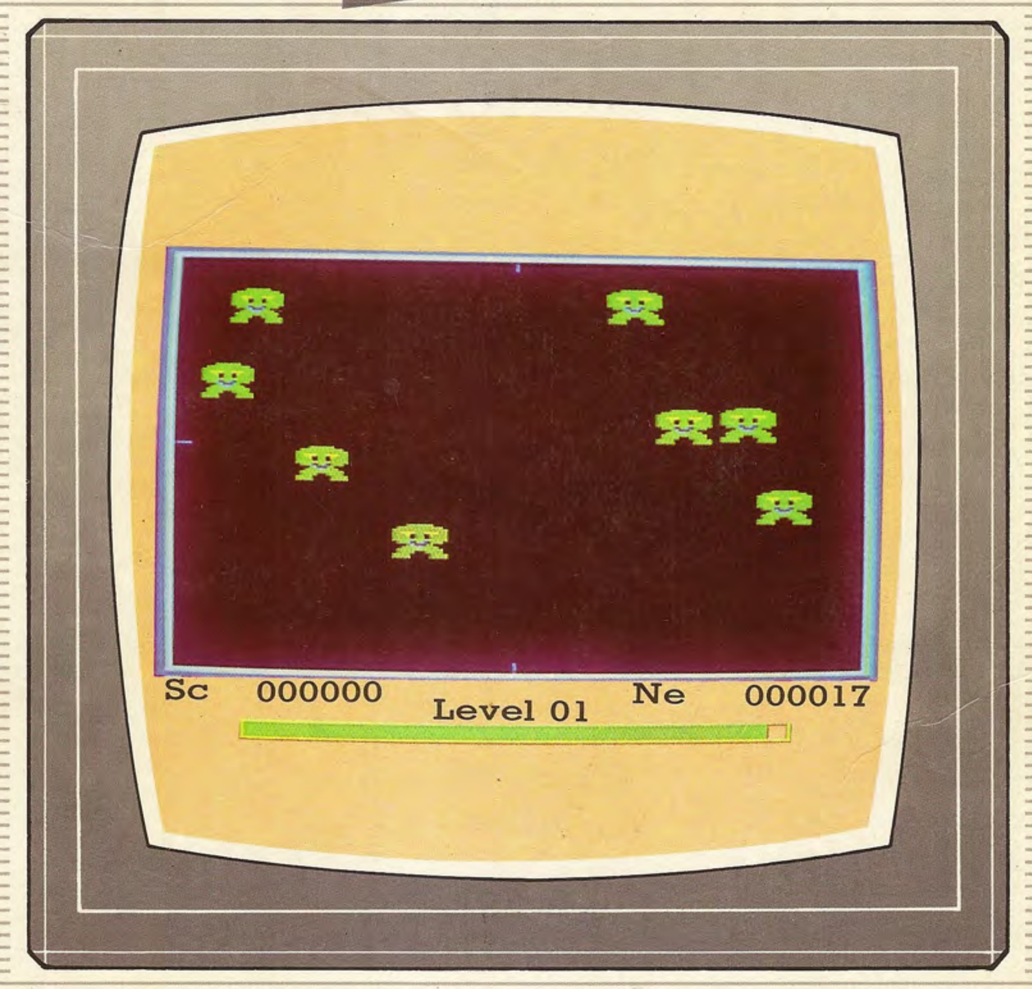
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 6

PASO A PASO



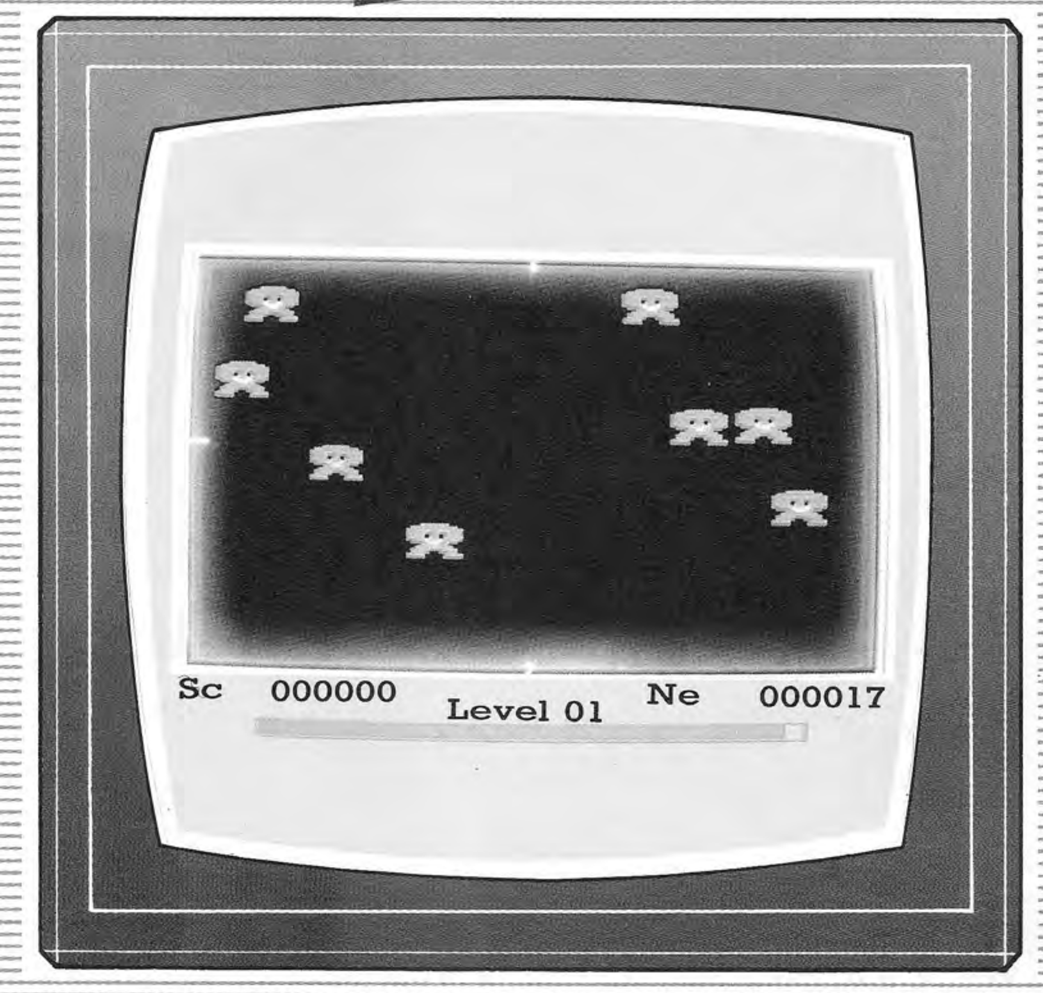
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 6

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomuniación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: Jesús Bocho, Licenciado en Informática. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Fernando Suero, Diplomado en Telecomunicación. OTROS LENGUAJES (Sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-079-0.
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Mayo, 1987.
P.V.P. Canarias: 335,–.

INDICE

INFORMATICA BASICA

EVALUACION DE ORDENADORES



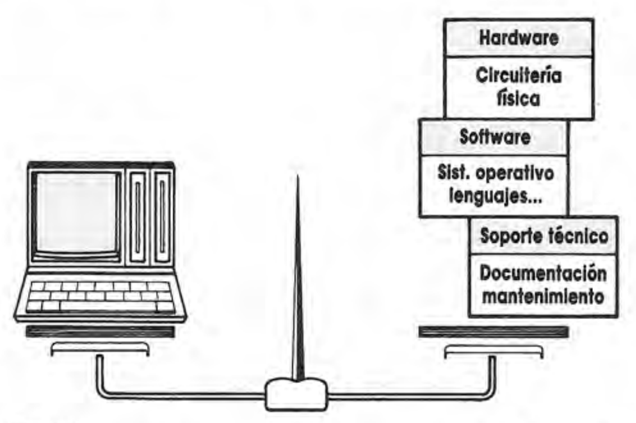


Evaluación del Hardware

ANTES de hablar de especificaciones técnicas vamos a aclarar una serie de conceptos básicos a los que haremos referencia más adelante.

Hardware

Se entiende por hardware el conjunto de componentes físicos que constituyen el ordenador. En anteriores capítulos hemos hecho una definición de ellos; sin embargo, vamos a recordarlos:



No sólo hay que sopesar en el análisis de un ordenador su estructura física; las posibilidades de programación y documentación también son decisivas.

El **procesador.** Es el elemento base de la parte física del ordenador. Su función es la de procesar los datos, para lo cual debe estar conectado, por un lado, a los periféricos de entrada/salida, y por otro, a la memoria de masa, los cuales se encargan de proporcionar al procesador la información a tratar.

La **memoria principal.** Su función es almacenar la información que produce el procesador al ir desarrollando las instrucciones de su programa. Esta información se podrá ir almacenando en zonas de almacenamiento permanente (memoria de masa) o bien ser borrada, al ejecutarse un nuevo programa.

Memorias de masa. Sirven fundamentalmente para almacenar:

— Ficheros de usuario.
— Los programas de aplicaciones del usuario.

El almacenamiento se puede realizar en cualquier soporte (cinta, disco, disquete, casete), pero se usa el disco preferentemente porque es más rápido que los anteriores.

A continuación se incluye un cuestionario de datos técnicos y económicos cuya recomendación se hace a los usuarios a la hora de requerir información técnica de los constructores.

1. **Para la unidad central de proceso**

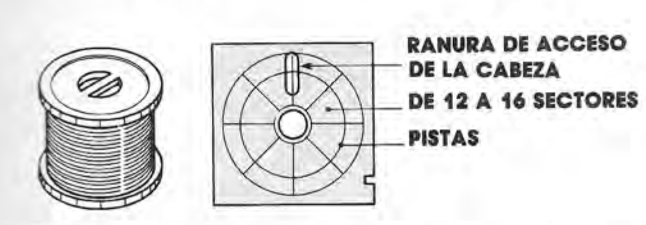
— Longitud de la palabra y de la instrucción.
— Características del repertorio de instrucciones.
— Descripción del sistema de protección de memoria.
— Descripción del sistema de direccionamiento.
— Descripción del sistema de interrupciones.

2. **Para la memoria principal**

— Capacidad propuesta.
— Capacidad máxima.
— Tipo de construcción.
— Corrección de errores.

3. **Unidades de discos**

— Capacidad por "dispack" en bytes.
— Velocidad de transferencia.
— Tiempo medio de acceso.
— Expresión de si puede servir como soporte del sistema operativo.




Tanto la capacidad de almacenamiento en "dis-pack" como las características de los disquetes utilizados son puntos importantes a tener en cuenta a la hora de elegir un sistema.

4. **Pantallas**

— Tamaño de la pantalla.
— Capacidad de la pantalla.
— Número de caracteres.
— Distintas intensidades de iluminación de pantalla.

Evaluación del software

Software

El software lo constituyen los elementos lógicos, frente al hadware, que lo forman los dispositivos físicos que conforman el ordenador. A la hora de comprar un ordenador, una elección adecuada del software es esencial para el buen aprovechamiento del equipo. La decisión vendrá influenciada, por un lado, por el software de base, y por otro, por el software de aplicaciones disponibles para el sistema.

Sistema operativo. Es el soporte básico que permite extraer las posibilidades del hardware y hacerlas asequibles al programador. El sistema operativo lo constituyen un conjunto de reglas establecidas en un ordenador para su correcta utilización. El sistema operativo debe asegurar la carga, supervisión y ejecución de los programas, el control de las operaciones de entrada/salida, la asignación de memoria, detección y corrección de errores, etc. Los sistemas operativos pueden trabajar en "batch" —ejecución de una o más tareas secuencialmente con alimentación previa de todos los datos—, en modo "interactivo" —que permite la entrada de datos durante la ejecución del proceso— en "tiempo real" —mediante el cual el sistema responde a las demandas externas según un concepto de prioridades— o en "tiempo compartido" —muchos usuarios pueden acceder al sistema simultáneamente y compartir todos sus recursos (memoria, ficheros, dispositivos...).

Lenguajes de programación. Son las herramientas que permiten elaborar los programas ejecutables por el procesador. Estos programas pueden formar parte de las aplicaciones de los usuarios o bien del sistema operativo.

Si se trata de lenguajes orientados al sistema operativo, éstos serán de bajo nivel y suponen una rápida y fácil ejecución. Si se trata de lenguajes orientados al usuario, conocidos como lenguajes de alto nivel, suponen un potente desarrollo y una mayor facilidad de utilización para el programador.

El ensamblador es un lenguaje orientado a la máquina y es importante para la preparación de otros programas. Se escribe en un formato simplificado y utiliza códigos nemotécnicos y direcciones de operandos simbólicas, produciendo un programa listo para su ejecución, una vez que las direcciones simbólicas hayan sido convertidas en otras de máquina, que son las únicas que el ordenador entiende.

En cuanto a los lenguajes orientados al usuario, deben incluir su correspondiente compilador, que es quien se encarga de traducir cada instrucción a lenguaje máquina. La adopción de uno o más lenguajes lleva consigo no sólo aumento de coste de ellos, sino también de los compiladores para cada uno. Los lenguajes simbólicos más conocidos son: destinados a gestión, como el RPG o el COBOL; otros de carácter científico, como el FORTRAN y el ALGOL, y otros de carácter general, como el BASIC.

Por último, es muy importante considerar también los programas de aplicación ofrecidos por el suministrador, tales como los "sorts" y los de mantenimiento de ficheros, y las aplicaciones estándar como: bases de datos, tratamiento de

textos, hojas electrónicas, aplicaciones gráficas, etc.



Cuestionario de software

1. **Sistemas operativos**

— Tipo de sistema operativo.
— Descripción de gestores: de entradas/salidas, de archivos, de sistemas de memoria virtual, de multiprogramación.

2. **Lenguajes**

— Lenguajes admisibles por la máquina.
— Versión de los lenguajes ofertados.
— Especificaciones a las que responde.





La elección del lenguaje apropiado facilitará la resolución de los problemas.

3. **Rutinas de utilidad y lenguajes de aplicación**

— Tipo de rutinas de utilidad y funciones que cumple.
— Tipos de aplicaciones que se ofertan con el sistema.
— Cantidad de memoria necesaria.



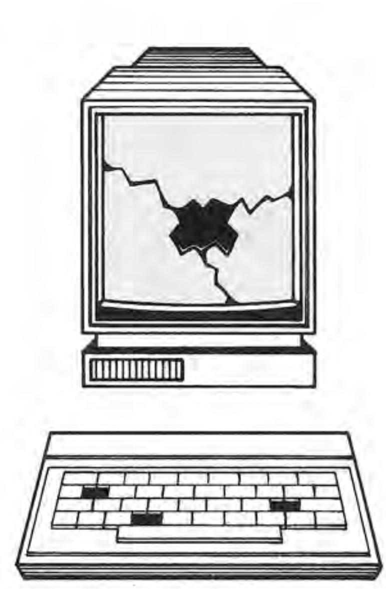
Otros aspectos a considerar

1. **Fecha de aparición en el mercado**

Indica la antigüedad del equipo, y como consecuencia, la vida que potencialmente le resta. Conviene fijarse que la antigüedad del sistema no sea superior a cuatro años. El usuario que adquiera un ordenador debe contar con que le deberá ser útil durante unos diez años, para lo cual deberá considerar detenidamente el tiempo de permanencia en el mercado.

2. **Asistencia técnica de sistemas**

Aunque el compromiso en los minis y micros no es tanto como el que se puede alcanzar en los grandes sistemas, su necesidad es importante igualmente. Los constructores están obligados a prestar un soporte de sistemas para implantar el software estándar. El usuario no debe abonar nada por el servicio, el cual no debe ser deducido del total de ayuda técnica que el constructor se comprometa a facilitar gratuitamente.





Compromiso de mantenimiento: no todos los constructores lo asumen por igual.

3. **Documentación**

El usuario debe contar siempre con una completa biblioteca de documentación referente al ordenador y al sistema operativo que utilice. Antes de firmar ningún contrato, el usuario debe conocer el coste de la documentación necesaria para la completa utilización de los recursos de su sistema.



MAQUINA Z-80

SPECTRUM, AMSTRAD, MSX



Descripción de las instrucciones



UNA vez vista la estructura general de la CPU y sus modos de direccionamiento entremos ahora en el estudio detallado de las instrucciones disponibles por el Z-80.

En el capítulo trataremos las instrucciones que versan sobre los movimientos de datos en memoria y registros. Es uno de los tipos de instrucciones más utilizados por los ordenadores, siendo el Z-80 uno de los microprocesadores más versátiles en el manejo de este tipo de instrucciones.



Movimientos de 8 y 16 bits

Estas instrucciones permiten copiar un dato de 8 ó 16 bits desde una posición de memoria, o de un registro, a otra posición de memoria, u otro registro.

Esta instrucción puede utilizarse con los diferentes tipos de direccionamiento explicados en el apartado anterior.

El formato de este tipo de instrucciones es el siguiente:

LD | 1.er OPERANDO | 2.° OPERANDO |



Fig. 1.

LD (de LoaD en inglés) significa cargar un dato.

El primer operando indica el destino donde se ha de cargar el dato que indica el segundo operando. Por tanto, el primer operando puede ser un registro o una dirección de memoria, ambos de 8 o de 16 bits, y el segundo puede ser un número binario, el contenido de un registro o el contenido de una dirección de memoria (de 8 o de 16 bits). Como puede verse en la tabla adjunta, debe tenerse bien claro que cuando se utiliza un dato de 16 bits, éste ocupa dos posiciones de memoria (bytes), y al dar una sola dirección, nos referimos a esa posición de memoria y a la siguiente (el Z-80 lo entiende así). En el caso de registros de 16 bits, nos referimos a la agrupación de dos registros de 8 bits, de la forma estándar, es decir, AF, BC, DE y HL, o de algún registro de 16 bits, como el puntero del stack (SP).

En el caso del Z-80 (al contrario de otros microprocesadores como el 6502), son prácticamente posibles todas las combinaciones. En la tabla adjunta pueden verse todas ellas:

Instrucciones de movimiento de bloques

Código mnemotécnico	Operación simbólica	Indicadores S Z H P/V N C	Códigos 76 543 210 Hex	N.º de bytes	N.º de ciclos M	N.º de estados T	Comentarios
EX DE,HL	DE↔HL	• • X • X • • •	11 101 011 EB	1	1	4	
EX AF,AF'	AF↔AF'	• • X • X • • •	00 001 000 08	1	1	4	
EXX	BC↔BC' DE↔DE' HL↔HL'	• • X • X • • •	11 011 001 D9	1	1	4	Intercambio del grupo de registros con el grupo alternativo
EX (SP),HL	H↔(SP+1) L↔(SP)	• • X • X • • •	11 100 011 E3	1	5	19	
EX (SP),IX	IX_H↔(SP+1) IX_L↔(SP)	• • X • X • • •	11 011 101 DD 11 100 011 E3	2	6	23	
EX (SP),IY	IY_H↔(SP+1) IY_L↔(SP)	• • X • X • • •	11 111 101 FD 11 100 011 E3	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	• • X 0 X ↕① 0 •	11 101 101 ED 10 100 000 A0	2	4	16	Carga (HL) en (DE), incrementa HL y DE y disminuye el contador BC
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 Repetir hasta que BC=0	• • X 0 X 0 0 •	11 101 101 ED 10 110 000 B0	2 2	5 4	21 16	Si BC≠0 Si BC=0
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	• • X 0 X ↕① 0 •	11 101 101 ED 10 101 000 A8	2	4	16	
LDDR	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1 Repetir hasta que BC=0	• • X 0 X 0 0 •	11 101 101 ED 10 111 000 B8	2 2	5 4	21 16	Si BC≠0 Si BC=0
CPI	A−(HL) HL←HL+1 BC←BC−1	↕ ↕③ X ↕ X ↕① 1 •	11 101 101 ED 10 100 001 A1	2	4	16	
CPIR	A−(HL) HL←HL+1 BC←BC−1 Repetir hasta que A=(HL) o BC=0	↕ ↕③ X ↕ X ↕② 1 •	11 101 101 ED 10 110 001 B1	2 2	5 4	21 16	Si BC≠0 y A≠(HL) Si BC=0 o A=(HL)
CPD	A−(HL) HL←HL−1 BC←BC−1	↕ ↕③ X ↕ X ↕① 1 •	11 101 101 ED 10 101 001 A9	2	4	16	
CPDR	A−(HL) HL←HL−1 BC←BC−1 Repetir hasta que A=(HL) o BC=0	↕ ↕③ X ↕ X ↕② 1 •	11 101 101 ED 10 111 001 B9	2 2	5 4	21 16	Si BC≠0 y A≠(HL) Si BC=0 o A=(HL)

NOTAS: ① P/V a 0 si el resultado de BC-1 es 0: P/V a 1 en caso contrario. ② P/V a 0 si se ha completado el recorrido; P/V a 1 en caso contrario. ③ Z a 1 si A = (HL); Z a 0 en caso contrario.


Fig. 2.

En este tipo de tablas, la primera columna indica la forma de escribir la instrucción en lenguaje ensamblador. En la segunda se explica simbólicamente su forma de operar. La tercera indica los cambios que se producirán en el registro de flags al ejecutar la instrucción. La siguiente indica la traducción a hexadecimal de las mismas. Y las demás indican otros parámetros de interés.

Como es de suponer, los paréntesis indican direccionamiento indirecto, y cuando a un dato se le suma IX o IY, indica indexación, como ya se explicó anteriormente.

Veamos esto con unos ejemplos:

1. LD (BC),A

Carga en la posición de memoria cuya dirección está en la dirección BC (2 Bytes = 16 bits = 1 dirección), el contenido del acumulador.

2. LD (IY+d),n

Carga en la dirección de memoria que se encuentra en otra posición resultante

de la suma del contenido del registro índice IY, con una dirección numérica (representada por *d)*, el valor *n*, por ejemplo:

LD (IY+$3FAB),3C

Carga en la dirección (IY+$3FAB) el número de 8 bits 3C. Es decir, en el byte IY+$3FAB y en el siguiente se encuentra la dirección donde cargar el dato.

3. Ahora con 16 bits:

LD (nn),(dd) como LD($3A18),($n31F)

La instrucción carga en la posición de memoria cuya dirección se encuentra en las direcciones *nn* y *nn+1*, el dato cuya dirección se encuentra en la dirección *dd.*



Intercambios entre registros

El Z-80 dispone de algunas instrucciones que permiten intercambiar los contenidos de algunos registros. Intercambiar no es lo mismo que cargar, ya que al cargar los dos registros implicados acaban con el mismo dato, mientras que en los intercambios sólo se les cambia de sitio.

Las instrucciones son las siguientes:

— EX DE,HL: Intercambia el contenido de los registros DE y HL.

— EX AF,AF': Cambia el registro AF con el equivalente del juego de registros complementarios.

— EXX: Esta instrucción cambia el resto de los registros con los del juego complementario.

Existen, además, otras instrucciones para intercambiar con el contenido de la pila del sistema, o stack, pero éstas se verán más adelante.

Instrucciones de movimiento de 8 bits

Código mnemotécnico	Operación simbólica	Indicadores S Z H P/V N C	Códigos 76 543 210 Hex	N.° de bytes	N.° de ciclos M	N.° de estados T	Comentarios
LD r,r'	r←r'	• • X • X • • •	01 r r'	1	1	4	r,r' Reg.
LD r,n	r←n	• • X • X • • •	00 r 110 ←n→	2	2	7	000 B
LD r,(HL)	r←(HL)	• • X • X • • •	01 r 110	1	2	7	001 C
LD r,(IX+d)	r←(IX+d)	• • X • X • • •	11 011 101 DD 01 r 101 ←d→	3	5	19	010 D 011 E 100 H
LD r,(IY+d)	r←(IY+d)	• • X • X • • •	11 111 101 FD 01 r 110 ←d→	3	5	19	101 L 111 A
LD (HL),r	(HL)←r	• • X • X • • •	01 110 r	1	2	7	
LD (IX+d),r	(IX+d)←r	• • X • X • • •	11 011 101 DD 01 110 r ←d→	3	5	19	
LD (IY+d),r	(IY+d)←r	• • X • X • • •	11 111 101 FD 01 110 r ←d→	3	5	19	
LD (HL),n	(HL)←n	• • X • X • • •	00 110 110 36 ←n→	2	3	10	
LD (IX+d),n	(IX+d)←n	• • X • X • • •	11 011 101 DD 00 110 110 36 ←d→ ←n→	4	5	19	
LD (IY+d),n	(IY+d)←n	• • X • X • • •	11 111 101 FD 00 110 110 36 ←d→ ←n→	4	5	19	
LD A,(BC)	A←(BC)	• • X • X • • •	00 001 010 0A	1	2	7	
LD A,(DE)	A←(DE)	• • X • X • • •	00 011 010 1A	1	2	7	
LD A,(nn)	A←(nn)	• • X • X • • •	00 111 010 3A ←n→ ←n→	3	4	13	
LD (BC),A	(BC)←A	• • X • X • • •	00 000 010 02	1	2	7	
LD (DE),A	(DE)←A	• • X • X • • •	00 010 010 12	1	2	7	
LD (nn),A	(nn)←A	• • X • X • • •	00 110 010 32 ←n→ ←n→	3	4	13	
LD A,I	A←I	↕ ↕ X 0 X IFF 0 •	11 101 101 ED 01 010 111 57	2	2	9	
LD A,R	A←R	↕ ↕ X 0 X IFF 0 •	11 101 101 ED 01 011 111 5F	2	2	9	
LD I,A	I←A	• • X • X • • •	11 101 101 ED 01 000 111 47	2	2	9	
LD R,A	R←A	• • X • X • • •	11 101 101 ED 01 001 111 4F	2	2	9	

NOTAS: r,r' representan cualquiera de los regstros A, B, C, D, E, H, L. IFF significa que el contenido (IFF) de la báscula de habilitación de interrupciones se carga en P/V. Para los diferentes símbolos que se emplean véase la tabla que figura al final del apéndice.



Fig. 3.

Instrucciones de movimiento de 16 bits

Código mnemotécnico	Operación simbólica	Indicadores S Z H P/V N C	Códigos 76 543 210 Hex	N.° de bytes	N.° de ciclos M	N.° de estados T	Comentarios
LD dd,nn	dd←nn	• • X • X • • •	00 dd0 001 ←n→ ←n→	3	3	10	dd Par 00 BC 01 DE 10 HL 11 SP
LD IX,nn	IX←nn	• • X • X • • •	11 011 101 DD 00 100 001 21 ←n→ ←n→	4	4	14	
LD IY,nn	IY←nn	• • X • X • • •	11 111 101 FD 00 100 001 21 ←n→ ←n→	4	4	14	
LD HL,(nn)	H←(nn+1) L←(nn)	• • X • X • • •	00 101 010 2A ←n→ ←n→	3	5	16	
LD dd,(nn)	dd_H←(nn+1) dd_L←(nn)	• • X • X • • •	11 101 101 ED 01 dd1 011 ←n→ ←n→	4	6	20	
LD IX,(nn)	IX_H←(nn+1) IX_L←(nn)	• • X • X • • •	11 011 101 DD 00 101 010 2A ←n→ ←n→	4	6	20	
LD IY,(nn)	IY_H←(nn+1) IY_L←(nn)	• • X • X • • •	11 111 101 FD 00 101 010 2A ←n→ ←n→	4	6	20	
LD (nn),HL	(nn+1)←H (nn)←L	• • X • X • • •	00 100 010 22 ←n→ ←n→	3	5	16	
LD (nn),(dd)	(nn+1)←dd_H (nn)←dd_L	• • X • X • • •	11 101 101 ED 01 dd0 011 ←n→ ←n→	4	6	20	
LD (nn),IX	(nn+1)←IX_H (nn)←IX_L	• • X • X • • •	11 011 101 DD 00 100 010 22 ←n→ ←n→	4	6	20	
LD (nn),IY	(nn+1)←IY_H (nn)←IY_L	• • X • X • • •	11 111 101 FD 00 100 010 22 ←n→ ←n→	4	6	20	
LD SP,HL	SP←HL	• • X • X • • •	11 111 001 F9	1	1	6	
LD SP,IX	SP←IX	• • X • X • • •	11 011 101 DD 11 111 001 F9	2	2	10	
LD SP,IY	SP←IY	• • X • X • • •	11 111 101 FD 11 111 001 F9	2	2	10	

NOTAS: dd es cualquiera de los pares de registros BC, DE, HL, SP. qq es cualquiera de los pares de registros AF, BC, DE, HL. $(par)_H$ y $(par)_l$ se refieren al byte alto y al byte bajo del par; por ejemplo, BC_l = C, AF_H = A.



Transferencia de bloques

Este conjunto de instrucciones permite mover bloques de memoria. Su utilidad es muy grande para implementar instrucciones tales como el tratamiento de tiras de caracteres (STRINGS).

— LDI: Mueve el contenido de la posición indicada por DE a la dirección indicada por HL. Después incrementa en uno los contenidos de DE y HL, con lo que prepara para la ejecución de la siguiente instrucción. Además, decrementa BC en una unidad.

— LDD: Es idéntica a la anterior, pero con la diferencia de que decrementa en una unidad los contenidos de DE y HL.

— LDIR: Esta instrucción ejecuta repetidamente la instrucción LDI hasta que el contenido de BC se hace 0. De esta forma puede moverse una gran cantidad de datos con una sola instrucción.

— LDDR: Esta es como la anterior, sólo que aquí la ejecución repetida es de la instrucción LDD, también hasta que BC llega a 0.



Búsqueda de bloques

Este conjunto de instrucciones permite buscar un determinado carácter dentro de un bloque. El carácter a buscar debe estar previamente en el acumulador (A).

— CPI: Compara el contenido del acumulador con el contenido de la dirección indicada por HL. Después incrementa HL en 1 y decrementa BC en uno.

— CPD: Es idéntica a la CPI, con la particularidad de que decrementa en una unidad HL.

— CPIR: Ejecuta repetidamente una instrucción CPI hasta que el contenido de HL es idéntico al acumulador o BC llega a cero. De esta forma puede buscarse dentro de un bloque un determinado dato.

— CPDR: Es igual que la anterior, pero con la ejecución repetida de la instrucción CPD.

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS



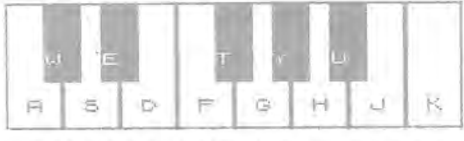
Programa: Organo electrónico



EL programa que aparece a continuación es un órgano electrónico (computarizado) para el SPECTRUM. Poco hay que decir sobre el programa aparte de que está escrito en código máquina y de que nos puede permitir el interpretar, mediante el teclado de nuestro ordenador, todas las melodías que nosotros queramos.

Sólo hay que llamar la atención sobre algunas líneas. El programa utiliza tres caracteres definidos. Las líneas en las cuales se encuentran dichos caracteres son la 60 y la 90. En estas líneas todos los caracteres que aparecen después de las comillas de la sentencia PRINT hay que escribirlos en modo GRAPHICS. El modo GRAPHICS se consigue pulsando a la vez las teclas CAPS SHIFT y el nueve (9). Una vez hecho esto, el cursor se nos aparecerá como una G parpadeante. Cuando tengamos el cursor en modo GRAPHICS, sólo tenemos que pulsar la tecla que corresponda según el programa. Aunque el carácter que aparezca por pantalla nos resulte extraño, no tenemos que preocuparnos, ése es el carácter que tiene que salir.

Una vez que el programa esté introducido en la memoria y todas las líneas DATA estén correctamente escritas, tras hacer RUN, la pantalla del ordenador se oscurecerá. Cuando esto ocurra no tienes por qué preocuparte, el ordenador está cargando todo el código máquina en su memoria. Una vez que esté cargado todo el código máquina, nos aparecerá el teclado de un piano en la pantalla. Este es el momento de empezar a utilizar el programa.



Las teclas que puedes utilizar para tocar música son las siguientes:

A - DO
W - DO sostenido
S - RE
E - Re sostenido
D - MI
F - FA
T - FA sostenido
G - SOL
Y - SOL sostenido
H - LA
U - LA sostenido
J - SI
K - DO (Una octava más arriba)

Cuando te canses de tocar el órgano sólo tienes que pulsar la tecla SPACE para volver al BASIC.

```
   1 REM ************************
   2 REM *(c)Ed. Siglo Cultural*
   3 REM *(c)1987              *
   4 REM ************************
   5 REM
  10 BORDER 1: PAPER 1: INK 7: C
LEAR 39999
  20 GO SUB 9000: REM COLOCA LA
SUBRUTINA EN CM
  30 GO SUB 8000: REM UDG
  40 REM DIBUJO DEL TECLADO
```

```
 50 FOR I=8 TO 12
 60 PRINT AT I,4;"C 88 88 BC 88
88 88 BC B"
 70 NEXT I
 80 FOR I=13 TO 15
 90 PRINT AT I,4;"C BC BC BC BC
BC BC BC B"
 100 NEXT I
 110 PLOT 32,111: DRAW 191,0
 120 PLOT 32,48: DRAW 191,0
 130 PRINT AT 0,4;"++ORGANO ELEC
TRONICO++"
 140 PRINT AT 14,5;"A";AT 14,8;"
S";AT 14,11;"D";AT 14,14;"F";AT
14,17;"G";AT 14,20;"H";AT 14,23;
"J";AT 14,26;"K"
 150 PRINT  INVERSE 1;AT 11,6;"W
";AT 11,9;"E";AT 11,15;"T";AT 11
,18;"Y";AT 11,21;"U"
 160 PRINT AT 21,0;"    ESPACIO-
VUELVE A BASIC         "
 200 RANDOMIZE USR 40000
 210 PAUSE 0
 220 STOP
8000 REM UDGS
8010 FOR I=0 TO 7
8020 POKE I+USR "A",129
8030 POKE I+USR "B",1
8040 POKE I+USR "C",128
8050 NEXT I
8060 RETURN
9000 REM CARGA LAS LINEAS DATA
9005 LET control=0
9010 FOR I=40000 TO 40341
9020 READ A: POKE I,A: LET CONTR
OL=CONTROL+A
9030 NEXT I
9040 IF CONTROL<>25879 THEN  PRI

NT "ERROR EN LOS DATAS:REVISE LA
S  LINEAS DESDE LA 9200 HASTA L
A  9540": STOP
9050 RETURN
9200 DATA 243,62,13,50,24,157,62
,127,219,254
9210 DATA 31,48,42,205,158,156,5
8,24,157,254
9220 DATA 13,40,16,185,40,24,205
,121,156,121
9230 DATA 254,13,32,10,50,24,157
,24,223,121
9240 DATA 254,13,40,218,50,24,15
7,205,121,156
9250 DATA 205,216,156,24,207,251
,201,58,24,157
9260 DATA 87,130,130,95,22,0,33,
25,157,25
9270 DATA 126,35,94,35,86,38,89,
111,126,238
9280 DATA 15,119,26,167,200,133,
111,126,238,15
9290 DATA 119,19,24,244,14,0,62,
253,219,254
9300 DATA 6,5,205,210,156,208,62
,191,219,254
9310 DATA 31,31,6,3,205,210,156,
208,62,251
9320 DATA 219,254,31,6,2,205,210
,156,208,31
9330 DATA 31,208,12,62,223,219,2
54,31,31,31
9340 DATA 6,2,205,210,156,201,31
,208,12,16
9350 DATA 251,201,58,24,157,33,2
41,156,6,0
9360 DATA 79,9,9,9,94,35,86,35,1
10,38
9370 DATA 0,235,205,181,3,243,20
1,106,6,5
9380 DATA 179,5,5,17,5,5,198,4,6
,61
9390 DATA 4,7,37,3,10,87,3,9,196
,3
9400 DATA 8,12,6,5,96,5,6,128,4,
7
9410 DATA 140,3,9,255,3,8,0,4,64
,157
9420 DATA 8,93,157,11,107,157,13
,64,157,17
9430 DATA 93,157,25,126,157,23,1
07,157,20,93
9440 DATA 157.6.83.157.9.83.157.
15,83,157
9450 DATA 21,83,157,18,83,157,1,
31,1,31
9460 DATA 1,31,1,31,1,31,1,1,30,
1
9470 DATA 1,30,1,1,0,1,31,1,31,1
9480 DATA 31,1,31,1,0,32,32,32,3
2,31
9490 DATA 1,1,30,1,1,30,1,1,0,1
9500 DATA 31,1,31,1,31,1,31,1,30
,1
9510 DATA 1,30,1,1,30,1,1,0,1,1
9520 DATA 30,1,1,30,1,1,30,1,1,3
0
9530 DATA 1,1,30,1,1,30,1,1,30,1
9540 DATA 1,0
```

Notas sobre el programa Organo electrónico

El programa está hecho de tal forma que cualquier persona con conocimientos de código máquina puede utilizar las rutinas que se encuentran dentro del mismo para sus propios programas.

Programa: Agenda telefónica

El siguiente programa es una agenda telefónica que nos permitirá tener almacenados en el disco de nuestro IBM o compatible toda la información que deseemos sobre ciertas personas.



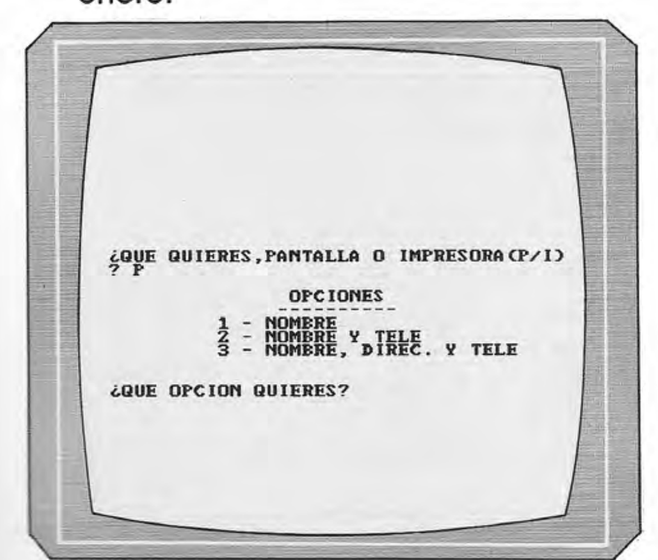

Menú del programa Gestor de teléfonos.

El programa nos permitirá tener un listín telefónico electrónico. Los datos que este programa puede almacenar son los siguientes:

— Nombre de la persona.
— Apellidos de la persona.
— Dirección en la cual vive.
— Ciudad.
— Código postal
— País.
— Teléfono.

Este programa nos permite realizar las siguientes funciones con las fichas que vayamos introduciendo en memoria:

— Introducir una persona nueva en el fichero.



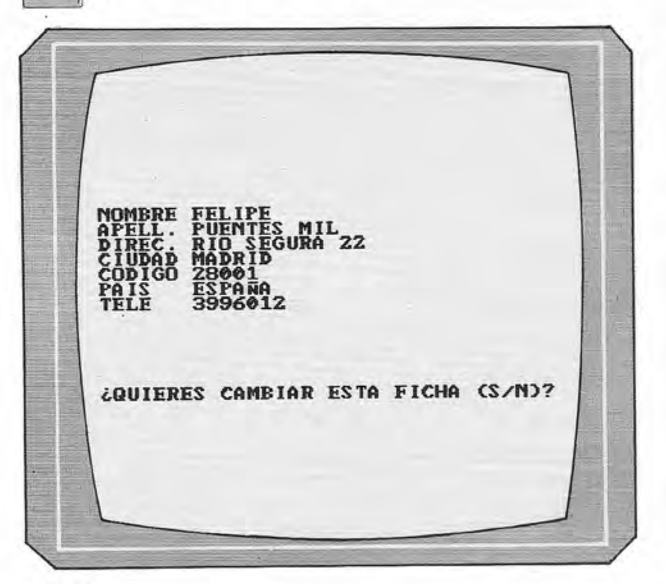

Menú de impresión.

— Buscar una persona.
— Cambiar los datos de una persona.
— Buscar todas las personas cuyo nombre empiece por una cierta letra.
— Borrar una persona del fichero.
— Listar todas las personas que hay almacenadas.

El programa también está capacitado para sacar por la impresora los datos de todo el listín telefónico que nosotros hayamos creado. Podemos sacar dicha impresión con caracteres normales o con letras comprimidas. Esta última opción está especialmente pensada para aquellos que quieran llevar una lista de todas las personas conocidas en la cartera.




Ejemplo de impresión.




El programa nos permite modificar las fichas.

Este programa está escrito para cualquier ordenador que sea compatible con el IMB pc, xt y at. También puede funcionar en el MSX y MSX2 con disco si se realizan los siguientes cambios:

```
100 REM ************************************
110 REM *                                  *
120 REM *          AGENDA TELEFONICA       *
130 REM *          -----------------       *
140 REM *                                  *
150 REM * POR: PETER BERGMANN              *
160 REM *                                  *
170 REM ************************************
180 REM *                                  *
190 REM *             VARIABLES            *
200 REM *             ---------            *
210 REM *                                  *
220 REM * REGISTRO                TEMPORAL *
230 REM *   S$      NOMBRE         W$, E$  *
240 REM *   A$     DIRECION        Y$      *
250 REM *   C$      CIUDAD         U$      *
260 REM *   Z$       ZONA          I$      *
270 REM *   P$       PAIS          D$      *
280 REM *   N$     TELEFONO        O$      *
290 REM *   L$  PUNTERO IZQUIERDA  L, F, LS*
300 REM *   R$  PUNTERO DERECHO    R       *
310 REM *   B$  PUNTERO ARRIBA     B       *
320 REM *                                  *
330 REM * OTRAS                            *
340 REM *   M   OPCION DE MENU             *
350 REM *   X$  FLAG PARA IZQ, DERC.       *
360 REM *   TN  NUMERO DEL PROXIMO REGISTRO*
370 REM *   TA  NUMERO DE REGISTROS        *
380 REM * U$,V$ PARA MANIPULAR STRING      *
390 REM *  Z,Q  NUMERO DE REGISTRO         *
400 REM *   M$  RESPUESTA (S/N)            *
410 REM *   I   CONTADOR                   *
420 REM * J$,C  RESPUESTA PARA CONTINUAR   *
430 REM *   D   FLAG                       *
440 REM *   H$  LETRA A BUSCAR             *
450 REM *   J$  PARA MANIPULACION DE STRING*
460 REM *   P   NUMERO DE PAGINA           *
470 REM *   G$  FLAG (P/I)                 *
480 REM *   U   OPCION DE PRINT            *
490 REM *   N   CONTADOR DE REGISTROS      *
500 REM *                                  *
510 REM ************************************
520 REM
530 REM **************************
540 REM * (c) Ed. Siglo Cultural *
550 REM * (c) 1987               *
560 REM **************************
570 REM
580 KEY OFF
590 CLS
600 PRINT "****************************************"
610 FOR I = 1 TO 19
620    PRINT  "*"; TAB(40); "*"
630 NEXT I
640 PRINT "****************************************"
650 LOCATE 8,12: PRINT "AGENDA TELEFONICA"
660 LOCATE 9,11: PRINT "------------------"
670 LOCATE 12,7: PRINT "(c) Ed. Siglo Cultural, 1987"
680 FOR I = 1 TO 1000
690 NEXT I
700 REM
```

```
710 REM *************************************
720 REM *        DEFINICION DE REGISTRO      *
730 REM *************************************
740 REM
750 OPEN "TELE.F" AS #1 LEN = 148
760 FIELD #1, 40 AS S$, 15 AS N$, 30 AS A$, 20 AS C$, 5 AS Z$, 30 AS P$, 2 AS L$
, 2 AS R$, 2 AS B$
770 REM*
780 REM ***  VER SI HAY REGISTROS  ***
790 REM
800 CLS
810 GET #1, 1
820 IF NOT EOF(1) THEN GOTO 900
830 LSET S$ = "M"
840 LSET N$ = MKI$(1)
850 LSET A$ = MKI$(1)
860 LSET L$ = MKI$(0)
870 LSET R$ = MKI$(0)
880 LSET B$ = MKI$(0)
890 PUT #1, 1
900 REM
910 REM *************************************
920 REM *                MENU                *
930 REM *************************************
940 REM
950 CLS
960 LOCATE 3,18: PRINT "MENU"
970 LOCATE 4,17: PRINT "------"
980 LOCATE 6,12: PRINT "1 - NUEVA PERSONA"
990 LOCATE 7,12: PRINT "2 - BUSCAR/CAMBIAR PERSONA"
1000 LOCATE 8,12: PRINT "3 - BUSCAR"
1010 LOCATE 9,12: PRINT "4 - BORRAR PERSONA"
1020 LOCATE 10,12: PRINT "5 - LISTAR TODAS"
1030 LOCATE 11,12: PRINT "6 - TERMINAR"
1040 LOCATE 14,1: PRINT "(QUE ELIGES? (1-6)";
1050 INPUT M
1060 IF (M < 1) OR (M > 6) GOTO 940
1070 ON M GOSUB 1170,2340,3060,3320,2760,1090
1080 GOTO 950
1090 REM
1100 REM ***  FIN DEL PROGRAMA  ***
1110 REM
1120 CLOSE #1
1130 CLS
1140 PRINT "ADIOS..."
1150 KEY ON
1160 END
1170 REM *************************************
1180 REM *         INTRODUCIR PERSONA        *
1190 REM *************************************
1200 REM
1210 CLS
1220 PRINT "PARA SALIR INTRODUZCA 'X' EN EL NOMBRE"
1230 PRINT
1240 LINE INPUT "NOMBRE: ";W$
1250 IF W$ = "X" THEN RETURN
1260 IF W$ = "" THEN GOTO 1210
1270 LINE INPUT "APELLIDOS: ";E$
1280 GOSUB 1520
1290 GOSUB 2230
1300 IF X$ = "D" GOTO 1450
1310 LINE INPUT "DIRECCION: ";Y$
1320 LINE INPUT "CIUDAD: ";U$
1330 LINE INPUT "CODIGO POSTAL: ";I$
1340 LINE INPUT "PAIS: ";D$
1350 LINE INPUT "NUMERO DE TELEFONO: ";O$
1360 GOSUB 2010
1370 GOSUB 2090
1380 GOSUB 1640
```

```
1390 TN = TN + 1
1400 TA = TA + 1
1410 LSET N$ = MKI$(TN)
1420 LSET A$ = MKI$(TA)
1430 PUT #1, 1
1440 GOTO 1170
1450 CLS
1460 PRINT "ESA FICHA YA EXISTE"
1470 PRINT:PRINT "PULSA UNA TECLA"
1480 H$=INPUT$(1)
1490 GOTO 1170
1500 END
1510 REM
1520 U$ = SPACE$(25)
1530 V$ = SPACE$(15)
1540 W$ = LEFT$(W$ + U$, 25)
1550 E$ = LEFT$(E$ + V$, 15)
1560 W$ = LEFT$(W$ + E$, 40)
1570 RETURN
1580 REM*
1590 REM* DISECT KEY
1600 REM*
1610 W$ = LEFT$(S$.25)
1620 E$ = RIGHT$(S$,15)
1630 RETURN
1640 REM
1650 REM* GET ROOT
1660 REM
1670 GET #1,1
1680 LET Q = 1
1690 TN = CVI(N$)
1700 TA = CVI(A$)
1710 RETURN
1720 REM*
1730 REM *** FICHA REPETIDA ***
1740 REM
1750 LET X$ = "D"
1760 REM
1770 RETURN
1780 REM
1790 REM *** COGER DERECHA ***
1800 REM
1810 R = CVI(R$)
1820 IF R = 0 THEN LET X$="R": GOTO 1850
1830 GET #1, R
1840 LET Q = R
1850 RETURN
1860 REM
1870 REM *** COGER IZQUIERDA ***
1880 REM
1890 L = CVI(L$)
1900 IF L = 0 THEN LET X$ = "L": GOTO 1930
1910 GET #1, L
1920 LET Q = L
1930 RETURN
1940 REM
1950 REM *** COGER ***
1960 REM
1970 B = CVI(B$)
1980 GET #1, B
1990 LET Q = B
2000 RETURN
2010 REM
2020 REM *** CAMBIAR ***
2030 REM
2040 TN = TN + 1
2050 IF X$ = "L" THEN LSET L$ = MKI$(TN)
2060 IF X$ = "R" THEN LSET R$ = MKI$(TN)
2070 PUT #1, Q
```

```
2080 RETURN
2090 REM
2100 REM *** GRABAR ***
2110 REM
2120 LSET S$ = W$
2130 LSET A$ = Y$
2140 LSET C$ = U$
2150 LSET Z$ = I$
2160 LSET P$ = D$
2170 LSET N$ = O$
2180 LSET L$ = MKI$(0)
2190 LSET R$ = MKI$(0)
2200 LSET B$ = MKI$(Q)
2210 PUT #1, TN
2220 RETURN
2230 REM
2240 REM *** BUSCAR FICHAS ***
2250 REM
2260 LET X$ = "N"
2270 GOSUB 1640
2280 WHILE X$ = "N"
2290   IF W$ > S$ THEN GOSUB 1780
2300   IF W$ < S$ THEN GOSUB 1860
2310   IF W$ = S$ THEN GOSUB 1720
2320 WEND
2330 RETURN
2340 REM
2350 REM ************************************
2360 REM *       LOCALIZAR UNA PERSONA      *
2370 REM ************************************
2380 REM
2390 CLS
2400 LOCATE 3,14: PRINT "BUSCAR PERSONA"
2410 LOCATE 5,1: PRINT "(NOMBRE";
2420 INPUT W$
2430 LOCATE 6,1: PRINT "(APELLIDOS";
2440 INPUT E$
2450 GOSUB 1510
2460 GOSUB 2230
2470 CLS
2480 IF X$ <> "D" THEN LOCATE 7,12: PRINT "FICHA NO ENCONTRADA": GOTO 2670
2490 GOSUB 1580
2500 LOCATE 2,2: PRINT "NOMBRE ";W$
2510 LOCATE 3,2: PRINT "APELL. ";E$
2520 LOCATE 4,2: PRINT "DIREC. ";A$
2530 LOCATE 5,2: PRINT "CIUDAD ";C$
2540 LOCATE 6,2: PRINT "CODIGO ";Z$
2550 LOCATE 7,2: PRINT "PAIS   ";P$
2560 LOCATE 8,2: PRINT "TELE   ";N$
2570 IF M <> 2 THEN GOTO 2730
2580 LOCATE 14,2: PRINT "(QUIERES CAMBIAR ESTA FICHA (S/N)";
2590 INPUT M$
2600 IF M$ = "S" THEN GOSUB 3980: GOTO 2490
2610 IF M$ <> "N" GOTO 2580
2620 LOCATE 14,2: PRINT SPACE$(38)
2630 LOCATE 14,2: PRINT "(QUIERES SACARLO POR IMPRESORA (S/N)";
2640 INPUT M$
2650 IF M$ = "S" THEN GOSUB 4140: GOTO 2680
2660 IF M$ <> "N" GOTO 2620
2670 IF M <> 2 THEN GOTO 2730
2680 LOCATE 14,2: PRINT SPACE$(38)
2690 LOCATE 14,2: PRINT "(QUIERES BUSCAR MAS (S/N)";
2700 INPUT M$
2710 IF M$ = "S" GOTO 2390
2720 IF M$ <> "N" GOTO 2690
2730 RETURN
2740 REM
```

```
2750 REM ***********************************
2760 REM *         IMPRIMIR ARBOL          *
2770 REM ***********************************
2780 REM
2790 GOSUB 4260
2800 LET P = 1
2810 CLS
2820 LOCATE 1,14: PRINT "LISTA DE NOMBRES"
2830 GOSUB 1640
2840 LET I = 1
2850 LET X$ = "N"
2860 WHILE X$ = "N"
2870   GOSUB 1870
2880 WEND
2890 L = CVI(L$)
2900 R = CVI(R$)
2910 B = CVI(B$)
2920 IF (Q <> 1) AND (M <> 3) THEN GOSUB 4500
2930 IF (M = 3) AND (Q <> 1) THEN GOSUB 3250
2940 IF I = TA THEN GOTO 3040
2950 I = I + 1
2960 LET X$ = "N"
2970 GOSUB 1790
2980 IF X$ = "N" GOTO 2860
2990 LET Z = Q
3000 GOSUB 1940
3010 R = CVI(R$)
3020 IF R = Z THEN GOTO 2990
3030 GOTO 2920
3040 PRINT: PRINT: PRINT "PULSA ENTER PARA CONTINUAR";:INPUT C
3050 RETURN
3060 REM
3070 REM ***********************************
3080 REM *            BUSCAR               *
3090 REM ***********************************
3100 REM
3110 CLS
3120 LOCATE 2,12: PRINT "BUSQUEDA POR UNA LETRA"
3130 LOCATE 4,1: PRINT "PRIMERA LETRA DEL NOMBRE";
3140 INPUT H$
3150 CLS
3160 LOCATE 2,16: PRINT "NOMBRES CON ";H$
3170 PRINT
3180 GOSUB 2830
3190 PRINT : PRINT
3200 PRINT "(QUIERES VER MAS (S/N):";
3210 INPUT M$
3220 IF M$ = "S" THEN GOTO 3100
3230 IF M$ <> "N" THEN GOTO 3200
3240 RETURN
3250 REM
3260 REM *** CHEQUEA LA PRIMERA LETRA ***
3270 REM
3280 J$ = LEFT$(S$,1)
3290 IF J$ = H$ THEN GOSUB 1580: PRINT W$;E$
3300 IF J$ > H$ THEN I = TA
3310 RETURN
3320 REM
3330 REM ************************************
3340 REM *         BORRAR REGISTRO          *.
3350 REM ************************************
3360 REM
3370 CLS
3380 LOCATE 3,11: PRINT "BORRAR UNA PERSONA"
3390 GOSUB 2410
3400 IF X$ = "D" THEN GOTO 3460
3410 LOCATE 12,1: PRINT "(QUIERES BORRAR MAS (S/N)";
3420 INPUT M$
```

```
3430 IF M$ = "S" GOTO 3370
3440 IF M$ <> "N" GOTO 3410
3450 RETURN
3460 LOCATE 14,2: PRINT "(ESTAS SEGURO (S/N)";
3470 INPUT M$
3480 IF M$ = "N" THEN CLS: GOTO 3410
3490 IF M$ <> "S" THEN GOTO 3460
3500 R = CVI(R$)
3510 L = CVI(L$)
3520 B = CVI(B$)
3530 LET D = 0
3540 IF (L <> 0) OR (R <> 0) THEN GOTO 3610
3550 GET #1, B
3560 F = CVI(L$)
3570 IF F = Q THEN LSET L$ = MKI$(0) ELSE LSET R$ = MKI$(0)
3580 PUT #1, B
3590 GOTO 3910
3600 REM
3610 IF R = 0 THEN GOTO 3710
3620 LET D = 1
3630 GET #1, B
3640 F = CVI(L$)
3650 IF F = Q THEN LSET L$ = MKI$(R) ELSE LSET R$ = MKI$(R)
3660 PUT #1, B
3670 GET #1, R
3680 LSET B$ = MKI$(B)
3690 PUT #1, R
3700 REM
3710 IF L = 0 THEN GOTO 3910
3720 IF D = 1 THEN GOTO 3820
3730 GET #1, B
3740 F = CVI(L$)
3750 IF F = Q THEN LSET L$ = MKI$(L) ELSE LSET R$ = MKI$(L)
3760 PUT #1, B
3770 GET #1, L
3780 LSET B$ = MKI$(B)
3790 PUT #1, L
3800 GOTO 3910
3810 REM
3820 GET #1, L
3830 LET W$ = S$
3840 LET LS = L
3850 GOSUB 2230
3860 IF X$ = "L" THEN LSET L$ = MKI$(LS) ELSE LSET R$ = MKI$(LS)
3870 PUT #1, Q
3880 GET #1, LS
3890 LSET B$ = MKI$(Q)
3900 PUT #1, LS
3910 CLS
3920 LOCATE 4,12: PRINT "FICHA BORRADA!"
3930 GOSUB 1640
3940 TA = TA - 1
3950 LSET A$ = MKI$(TA)
3960 PUT #1, 1
3970 GOTO 3410
3980 REM
3990 REM ***********************************
4000 REM *      CAMBIAR EL REGISTRO        *
4010 REM ***********************************
4020 REM
4030 LOCATE 14,2: PRINT SPACE$(38)
4040 LOCATE 14,2: PRINT "(QUE QUIERES CAMBIAR (D,C,Z,P,T)";
4050 INPUT M$
4060 IF M$ = "D" THEN LOCATE 4,9: LINE INPUT Y$: LSET A$ = Y$
4070 IF M$ = "C" THEN LOCATE 5,9: LINE INPUT U$: LSET C$ = U$
4080 IF M$ = "Z" THEN LOCATE 6,9: LINE INPUT I$: LSET Z$ = I$
4090 IF M$ = "P" THEN LOCATE 7,9: LINE INPUT D$: LSET P$ = D$
4100 IF M$ = "T" THEN LOCATE 8,9: LINE INPUT O$: LSET N$ = O$
```

```
4110 PUT #1, Q
4120 CLS
4130 RETURN
4140 REM
4150 REM ******************************
4160 REM *          IMPRIMIR          *
4170 REM ******************************
4180 REM
4190 GOSUB 4890
4200 LPRINT W$;E$
4210 LPRINT A$
4220 LPRINT C$;"  ";Z$
4230 LPRINT P$
4240 LPRINT N$
4250 RETURN
4260 REM
4270 REM ***********************************
4280 REM *   PANTALLA O IMPRESORA ????     *
4290 REM ***********************************
4300 REM
4310 CLS
4320 LOCATE 2,1: PRINT "(QUE QUIERES, PANTALLA OR IMPRESORA (P/I)";
4330 INPUT G$
4340 IF (G$ <> "P") AND (G$ <> "I") THEN GOTO 4310
4350 LOCATE 5,16: PRINT "OPCIONES"
4360 LOCATE 6,15: PRINT "----------"
4370 LOCATE 7,10: PRINT "1 - NOMBRE"
4380 LOCATE 8,10: PRINT "2 - NOMBRE Y TELE"
4390 LOCATE 9,10: PRINT "3 - NOMBRE, DIREC. Y TELE"
4400 LOCATE 12,1: PRINT "(QUE OPCION QUIERES";
4410 INPUT U
4420 IF (U < 1) OR (U > 3) THEN GOTO 4400
4430 LOCATE 14,1: PRINT "(ESTAS SEGURO (S/N)";
4440 INPUT M$
4450 IF M$ = "N" THEN GOTO 4310
4460 IF M$ <> "S" THEN GOTO 4430
4470 LET N = 0
4480 IF G$ = "I" THEN GOSUB 4890
4490 RETURN
4500 REM
4510 REM ******************************
4520 REM *        IMPRIMIR            *
4530 REM ******************************
4540 REM
4550 GOSUB 1580
4560 IF G$ = "I" THEN GOTO 4710
4570 PRINT W$;E$
4580 IF U = 3 THEN PRINT A$: PRINT C$,Z$: PRINT P$
4590 IF U > 1 THEN PRINT N$
4600 PRINT
4610 N = N + 1
4620 IF (U = 1) AND (N/P = 10) THEN GOTO 4660
4630 IF (U = 2) AND (N/P = 7) THEN GOTO 4660
4640 IF (U = 3) AND (N/P = 3) THEN GOTO 4660
4650 GOTO 4880
4660 P = P + 1
4670 PRINT: PRINT "PULSA ENTER ";
4680 J$ = INPUT$(1)
4690 CLS
4700 GOTO 4880
4710 REM
4720 REM ******************************
4730 REM *     SALIDA POR IMPRESORA   *
4740 REM ******************************
4750 REM
4760 LPRINT W$;E$
4770 IF U = 3 THEN LPRINT A$: LPRINT C$,Z$: LPRINT P$
4780 IF U > 1 THEN LPRINT N$
```

```
4790 LPRINT
4800 N = N + 1
4810 IF (U = 1) AND (N/P = 30) THEN GOTO 4850
4820 IF (U = 2) AND (N/P = 20) THEN GOTO 4850
4830 IF (U = 3) AND (N/P = 10) THEN GOTO 4850
4840 GOTO 4880
4850 P = P + 1
4860 LPRINT CHR$(12)
4870 CLS
4880 RETURN
4890 REM
4900 REM ********************************
4910 REM *      PREPARAR IMPRESORA      *
4920 REM ********************************
4930 REM
4940 LPRINT CHR$(27)+CHR$(64)
4950 CLS
4960 LOCATE 2,1: PRINT "(QUIERES IMPRESION GRANDE O PEQUENIA (G/P)";
4970 INPUT M$
4980 IF M$ = "P" THEN LPRINT CHR$(27)+CHR$(15)
4990 IF M$ = "G" THEN LPRINT CHR$(18)
5000 IF (M$ <> "P") AND (M$ <> "G") THEN GOTO 4950
5010 CLS
5020 LOCATE 2,1: PRINT "COLOCA EL PAPEL Y PULSA ENTER";
5030 J$ = INPUT$(1)
5040 RETURN
```

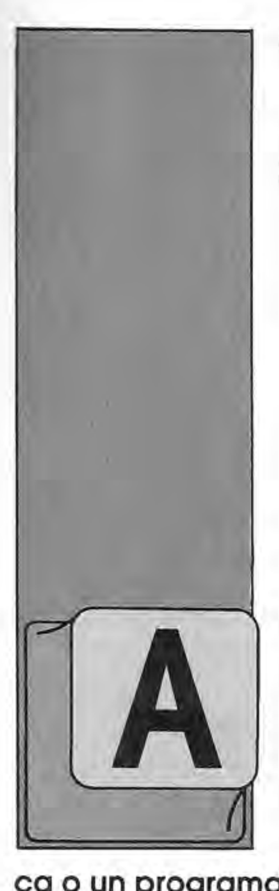
650 LOCATE 12,8:PRINT "AGENDA TELEFONICA"
660 LOCATE 11,9:PRINT "-----------------"
670 LOCATE 7,12:PRINT "(c) Ed. Siglo Cultural, 1987"
960 LOCATE 18,3:PRINT "MENU"
970 LOCATE 17,4:PRINT "------"
980 LOCATE 12,6:PRINT "1 - NUEVA PERSONA"
990 LOCATE 12,7:PRINT "2 - BUSCAR/CAMBIAR PERSONA"
1000 LOCATE 12,8:PRINT "3 - BUSCAR"
1010 LOCATE 12,9:PRINT "4-BORRAR PERSONA"
1020 LOCATE 12,10:PRINT "5 - LISTAR TODAS"
1030 LOCATE 12,11:PRINT "6 - TERMINAR"
1040 LOCATE 1,14:PRINT "(QUE ELIGES? (1-6) ";
2400 LOCATE 14,3:PRINT "BUSCAR PERSONA"
2410 LOCATE 1,5:PRINT "(NOMBRE";
2430 LOCATE 1,6:PRINT "(APELLIDOS ";
2510 LOCATE 2,3:PRINT "APELL. ";E$
2520 LOCATE 2,4:PRINT "DIREC. ";A$
2530 LOCATE 2,5:PRINT "CIUDAD ";C$
2540 LOCATE 2,6:PRINT "CODIGO ";Z$
2550 LOCATE 2,7:PRINT "PAIS ";P$
2560 LOCATE 2,8:PRINT "TELE ";N$
2580 LOCATE 2,14:PRINT "(QUIERES CAMBIAR ESTA FECHA? (S/N)";
2620 LOCATE 2,14:PRINT SPACE$(38)
2630 LOCATE 2,14:PRINT "(QUIERES SACARLO POR IMPRESORA? (S/N) ";
2680 LOCATE 2,14:PRINT SPACE$(38)
2690 LOCATE 2,14:PRINT "(QUIERES BUSCAR MAS? (S/N) ";
2820 LOCATE 14,1:PRINT "LISTA DE NOMBRES"
3120 LOCATE 12,2:PRINT "BUSQUEDA POR UNA LETRA"
3130 LOCATE 1,4:PRINT "PRIMERA LETRA DEL NOMBRE";
3160 LOCATE 16,2:PRINT "NOMBRES CON ";H$

```
3380 LOCATE 11,3:PRINT "BORRAR UNA PERSONA"
3410 LOCATE 1,12:PRINT "(QUIERES BORRAR MAS? (S/N) »;
3460 LOCATE 2,14:PRINT "(ESTAS SEGURO? (S/N) ";
3920 LOCATE 12,4:PRINT "FICHA BORRADA!"
4030 LOCATE 2,14:PRINT SPACE$(39)
4040 LOCATE 2,14:PRINT "(QUE QUIERES CAMBIAR (D,C,Z,P,T) ";
4060 IF M$="D"THEN LOCATE 9,4:LINE INPUT Y$:LSET A$=Y$
4070 IF M$="C"THEN LOCATE 9,5:LINE INPUT U$:LSET C$=U$
4080 IF M$="Z"THEN LOCATE 9,6:LINE INPUT I$:LSET Z$=I$
4090 IF M$="P"THEN LOCATE 9,7:LINE INPUT D$:LSET P$=D$
4100 IF M$="T"THEN LOCATE 9,8:LINE INPUT O$:LSET N$=O$
4320 LOCATE 1.2:PRINT "(QUE QUIERES, PANTALLA O IMPRESORA (P/I) ";
4350 LOCATE 16,5:PRINT "OPCIONES"
4360 LOCATE 15,5:PRINT "----------"
4370 LOCATE 10,7:PRINT "1 - NOMBRE"
4380 LOCATE 10,8:PRINT "2 - NOMBRE Y TELE."
4390 LOCATE 10,9:PRINT "(QUE OPCION QUIERES? ";
4400 LOCATE 1,12:PRINT "(QUE OPCION QUIERES? »;
4430 LOCATE 1,14:PRINT "(ESTAS SEGURO? (S/N) ";
4960 LOCATE 1,2:PRINT "(QUIERES IMPRESION GRANDE O PEQUEÑA (G/P) ";
5020 LOCATE 1,2:PRINT "COLOCA EL PAPEL Y PULSA ENTER";
```

Este programa aparecerá dentro de pocos tomos en versiones para el COMMODORE, AMSTRAD y SPECTRUM.

TECNICAS DE ANALISIS

SEGURIDADES EN LOS PROCESOS



DEMAS del control sobre los datos (para obviar la introducción de errores y asegurar la congruencia de los resultados), en el análisis de una aplicación informática o un programa conviene introducir algunos mecanismos de seguridad en el conjunto de los ficheros de datos a procesar.

En efecto, en todos los dispositivos físicos de soporte magnético de datos se pueden producir errores y accidentes que provocarán la destrucción de los datos o, al menos, dificultades en su lectura.

La situación es ligeramente diferente en el caso de manejo de cintas magnéticas y de discos fijos. Los disquetes magnéticos participan de ciertas características de ambos medios.

En el caso de las cintas magnéticas es usual prever varios procedimientos de seguridad.

a) Control de validez de los datos leídos: hay que establecer un procedimiento de reintentos de lectura y de información al usuario en caso de producirse errores. Se pueden fijar otros controles (de longitud de datos obtenidos, de secuencia de registros, etc.) para asegurar la correcta lectura de los datos.

b) Establecimiento de una copia de seguridad para un fichero, cuando la repetición del proceso de obtención de ese archivo es larga o complicada, cuando los datos deben ser utilizados en varios procesos, etc. Hay que evaluar en este caso el coste de inmovilización de las copias frente a la posibilidad de repetición de los procesos, los problemas de tiempo que pueden surgir, etc.

c) A veces, incluso, se establece un sistema de varias copias (normalmente tres: «abuelo-padre-hijo») cuando es importante el fichero a conservar o cuando el proceso se repite con mucha frecuencia.

d) Reducir los tiempos de realización en los procesos muy largos, previendo su fraccionamiento y estableciendo puntos de control y relanzamiento. Para ello hay que definir unos momentos en los que el proceso se detenga y en los que se obtenga una imagen de la situación (posición de los ficheros que se están procesando, estado de las variables, imagen de la memoria principal, etc.). De este modo, si en el intervalo hasta el siguiente punto de control se produce una anomalía, es posible relanzar el proceso desde el punto de control anterior en vez de arrancar la aplicación desde su comienzo. En estos puntos de relanzamiento se suelen establecer controles para detectar posibles anomalías no detectables por simple observación exterior del desarrollo del proceso y que no producen un mensaje de error del sistema.

e) Además de la definición de los sistemas de seguridad indicados, el analista deberá proveer normas detalladas de actuación de todas las personas involu-

cradas en los trabajos, de forma que se actúe de un modo preciso para la detección y eliminación o corrección de los errores que se puedan producir.

En el caso de los discos magnéticos se pueden producir, además, algunos otros errores o accidentes debidos a la propia estructura del dispositivo: detección del disco en el momento de escritura con destrucción física del soporte, borrado de etiquetas o índices de los ficheros etcétera, incluso problemas de polvo, al que son bastante sensibles.

Para evitar estas dificultades (o minimizarlas en la medida de lo posible) se recomienda:

a) Limitar la importancia de las correcciones a realizar si se produce una destrucción de información, mediante la división del fichero en bloques lógicos que representen porciones limitadas de los ficheros completos.

b) Prever un código incluido en el fichero; código que deberá ser analizado antes de la cumplimentación de cualquier orden de escritura.

c) Limitar los accesos de modificación de los ficheros básicos, mediante agrupación de las actualizaciones del archivo, en la medida de lo posible.

d) Distribuir los ficheros en varios dispositivos físicos o instalar en un dispositivo los índices de acceso y otros elementos de control y en otro diferente los datos. Por este procedimiento, además, se puede mejorar el tiempo de acceso a los archivos básicos, dependiendo del sistema físico (hardware) sobre el que se esté trabajando.

e) Control del momento en que se ha producido la anomalía (error, detención del dispositivo, etc.) respecto del momento en que se ha escrito o se deberá escribir, para deducir de ello las fases a cumplimentar en el proceso de reconstrucción del fichero y en el procedimiento de relanzamiento.

f) Creación de ficheros de seguridad parciales que se elaboren automáticamente en ciertos momentos del proceso o cuando se ha producido un número de accesos predefinido.

g) Establecimiento de ficheros de almacenamiento de actualizaciones para simplificar el relanzamiento.

h) Control de volumen de los ficheros para coadyuvar a los mecanismos de seguridad del sistema y prever áreas de expansión y reordenación de los archivos.

i) Prever y definir cuidadosamente los procesos de depuración y reorganización de los ficheros para minimizar estas tareas y eliminar posibles fuentes de problemas.

Ademas de estos mecanismos concretos de seguridad indicados, en la mayoría de las aplicaciones será posible, e incluso conveniente, establecer otras seguridades y controles que vendrán indicadas por las características de las tareas a desarrollar.

Conviene hacer un examen de los procesos desde este punto de vista y detectar las tareas especialmente críticas, extremando en ellas el número y sofisticación de las seguridades a definir e incluso, en algunos casos, estableciendo que los programas emitan mensajes de advertencia a los operadores.



TECNICAS DE PROGRAMACION

TIPOS DE DATOS

POR el momento hemos hablado bastante de estructuras de datos. En efecto, los escalares, las series o vectores y las tablas o matrices constituyen en conjunto la estructura de más del 99 por 100 de los datos que uno suele encontrarse en los programas de ordenador. Es verdad que existen otras estructuras: tablas rectangulares de tres o más dimensiones, listas, árboles y grafos que pueden tener gran utilidad en aplicaciones concretas, especialmente en lenguajes especializados (como LISP) o próximos al lenguaje de la máquina (como C). Describiremos más adelante estas otras estructuras.

Además de una estructura determinada, los datos que proporcionamos a un programa pueden pertenecer también a varios tipos diferentes. El tipo de un dato nos indica qué clase de información representa ese dato y qué operaciones podemos realizar con él. Por ejemplo, un dato de un programa podría ser la palabra «VERTEBRA». Otro dato del mismo programa podría ser el número 25. En el primer caso tenemos información alfabética con la que podemos realizar operaciones como las siguientes (entre otras muchas posibles):

— Ponerla por orden alfabético, lo que nos daría el resultado «ABEERRTV».

— Compararla con otra palabra, lo que nos daría el resultado «son iguales» o «son desiguales».

— Extraer los tres primeros caracteres, lo que nos daría el resultado «VER».

— Extraer dos caracteres a partir del cuarto, lo que nos daría el resultado «TE».

— Invertir el orden de sus letras, lo que nos dará el resultado «ARBETREV».

— Imprimirla por la pantalla.

En el caso del número 25 tenemos información numérica, con la que podemos realizar operaciones como las siguientes (también entre otras muchas):

— Sumarle el número 3, lo que nos daría el resultado 28.

— Multiplicarla por 2, lo que nos daría el resultado 50.

— Pasarla al sistema de numeración binario, lo que nos daría el resultado «11001».

— Compararla con otro número, lo que nos daría el resultado «son iguales», «son desiguales», «el primero es mayor» o «el segundo es mayor».

— Imprimirla por la pantalla.

Como se ve, algunas de las operaciones que podemos ralizar con la información alfabética (como ordenar por orden alfabético o extraer cierto número de caracteres a partir de uno dado) no son aplicables a la información numérica. Por el contrario, algunas de las operaciones que podemos realizar con datos numéricos (como la suma, la multiplicación o el cambio de sistema de numeración) no son aplicables a los datos alfabéticos. Por último, existen operaciones (como imprimir en la pantalla o comparar) que pueden realizarse indistintamente con ambos tipos de datos.

En las próximas páginas veremos con más detalle ambos tipos principales de datos, junto con sus posibles subdivisiones y las operaciones en que pueden tomar parte. Después mencionaremos algunos tipos de datos más, que se estudiarán con más detalle en otra parte de esta obra.



Datos numéricos

No cabe duda de que la información numérica desempeña un papel importantísimo en los programas de ordenador. Sin embargo, el mero hecho de poder operar con ella presenta cierto número de problemas, relativamente complejos, que vamos a describir sin entrar demasiado en detalle.

La información de cualquier tipo está contenida en la memoria de las computadoras en forma binaria, mediante circuitos eléctricos que pueden estar abiertos o cerrados, o que pueden tener tensiones altas o bajas, en forma de magnetizaciones norte o sur. Es decir, mediante sistemas capaces de tomar solamente dos estados. Por conveniencia, a uno de los estados se le llama «cero» y al otro «uno». Un dato cualquiera viene representado siempre, en definitiva, por una sucesión de estos elementos binarios. Dicho de otro modo, por una sucesión de ceros y unos.

Es evidente, por consiguiente, que existe una forma natural de representar la información numérica dentro de la memoria de un ordenador: el sistema binario de numeración. Como se sabe, los números se representan en este sistema utilizando únicamente dos cifras diferentes: el cero y el uno. Como comparación, pensemos que en el sistema decimal de numeración, al que estamos acostumbrados, los números se representan utilizando diez cifras diferentes.

¿Cómo construimos los números en el sistema decimal? Si los enumeramos todos sucesivamente, veremos que se construyen tomando primero una sola cifra y haciendo variar ésta del cero al nueve. Cuando se nos han acabado las cifras, ponemos un uno en la segunda cifra (la de las decenas) y volvemos a repetir el proceso de variar la primera cifra del cero al nueve. A continuación, sumamos una unidad a la segunda cifra y volvemos otra vez a variar la primera, y así sucesivamente. Cada vez que llegamos al final del recorrido de una cifra determinada, añadimos una unidad a la cifra siguiente hacia la izquierda y volvemos al cero en la cifra anterior (situada a su derecha).

Exactamente lo mismo se hace en el caso del sistema binario de numeración pero aquí no disponemos de diez cifras distintas (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), sino tan sólo de dos (0 y 1), por lo que las cifras de los números van extendiéndose hacia la izquierda mucho más de prisa. Veamos en la siguiente tabla cómo van generándose sucesivamente los números binarios, aplicando el método anterior. Como comparación, hemos colocado el equivalente decimal en la columna de la izquierda de la tabla:

N.° decimal	N.° binario	N.° decimal	N.° binario
0	0	10	1010
1	1	11	1011
2	10	12	1100
3	11	13	1101
4	100	14	1110
5	101	15	1111
6	110	16	10000
7	111	17	10001
8	1000	18	10010
9	1001	19	10011

De la misma forma anterior podrían generarse los números sucesivos de cualquier otra base: la ternaria (basada en las cifras 0, 1, 2), la octal (basada en las cifras 0, 1, 2, 3, 4, 5, 6, 7), la hexadecimal (en un sistema de dieciséis cifras, que usualmente se representan con los símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F), etc.

Pues bien: como hemos dicho, la forma «natural» de representar los números en un ordenador es en el sistema binario de numeración. Sin embargo, la forma «natural» de verlos en la pantalla o introducirlos en el teclado es el sistema decimal, que al ordenador le resulta extraño. Es preciso, por tanto, convertirlos de un sistema a otro.

Normalmente no tenemos que preocuparnos de realizar este tipo de conversiones, pues existen programas de ordenador que nos las hacen automáticamente.

Los traductores de los lenguajes de programación (compiladores o intérpretes) realizan esta función, entre otras muchas. Por esta razón podemos olvidarnos casi siempre del sistema binario de numeración y trabajar como si la máquina «entendiera» nuestros números decimales, aun cuando las operaciones se realizarán siempre dentro de ella en el sistema binario.

Dentro de los ordenadores hay un elemento, que se llama el microprocesador, que, entre otras cosas, está encargado de realizar las operaciones aritméticas con los datos numéricos. Para ello, dispone de una serie de «instrucciones de la máquina» elementales, capaces de realizar operaciones aritméticas sencillas (sumas, restas, multiplicaciones, divisiones) entre parejas de números representados en el sistema binario mediante tensiones eléctricas altas o bajas. Ocurre, sin embargo, que estos números tienen que tener una longitud determinada, pues no podemos construir microprocesadores que operen con números de cualquier longitud, ya que serían demasiado complejos. Por consiguiente, al diseñar el microprocesador se decide siempre cuántas cifras han de tener los números binarios con los que sea capaz de operar. En un microprocesador determinado se pueden elegir una o más de estas longitudes, pero siempre en un número fijo y determinado. En general, se operará con números binarios que tengan un número de cifras múltiplo de ocho: las longitudes más frecuentes son ocho, dieciséis, treinta y dos y sesenta y cuatro.

¿Qué ocurre si queremos operar con un número binario de cinco cifras en un microprocesador que sólo puede operar con números de ocho cifras? No hay problema. En todos los sistemas de numeración, los ceros a la izquierda no tienen valor alguno. Así, en el sistema decimal al que estamos acostumbrados, sabemos que 12345 representa el mismo número que 00012345. Lo mismo sucede en el sistema binario. Así, si queremos operar con el número binario equivalente al decimal 17 (10001) y debemos extenderlo a ocho cifras, bastará con añadirle tres ceros a la izquierda, convirtiéndolo en 00010001.

El problema se presenta si queremos operar con números binarios de nueve cifras en un microprocesador que sólo puede operar con números de ocho cifras. Por ejemplo: sea el número decimal 511, cuya representación binaria es 111111111. Es evidente que no podremos trabajar con él directamente, pues nos sobra una cifra. Siempre es posible construir un programa que parta el número en varios trozos (dos, en este caso) y realice las operaciones en varios pasos, pero esto es preciso programarlo. En cambio, si el microprocesador tuviera también la posibilidad de operar directamente con números binarios de dieciséis cifras (además de con los de 8) bastaría con añadir al número anterior siete ceros a la izquierda para resolver totalmente el problema.

¿A dónde vamos a parar con todo esto? Al hecho de que, cualquiera que sea nuestro ordenador, habrá siempre un límite a los números enteros con los que pueda operar directamente, que dependerá del número de cifras binarias elegido en el momento de diseñarlo. Todo esto se complica por el hecho de que generalmente se introducen convenios para representar también con cifras binarias los números negativos (como el sistema del complemento a dos), pero no vamos a entrar en ello. Bastará con ver, en la tabla siguiente, los límites correspondientes a las distintas longitudes utilizadas normalmente, tanto en el caso de números estrictamente positivos como de números positivos y negativos indistintamente. Los límites vienen dados en la tabla en el sistema decimal de numeración.

N.º cifras binarias	N.os positivos	N.os positivos y negativos
8	0 a 255	−128 a 127
16	0 a 65535	−32768 a 32767
32	0 a 4294967295	−2147483648 a 2147483647

Por supuesto, hasta ahora hemos hablado únicamente de números enteros. Pero es también muy frecuente que deseemos realizar operaciones con números decimales, como los siguientes:

2,5
0,0000000001
123,45

o con números enteros que se salgan de los límites con los que puede operar nuestro ordenador. También es posible que, aunque los números con los que operamos sean enteros y caigan dentro de los límites válidos, el resultado de la operación no lo sea. Veamos un ejemplo:

Sea un ordenador cuyo microprocesador puede realizar operaciones con números binarios de dieciséis cifras. Supongamos que acepta números negativos. De acuerdo con la tabla anterior, esto significa que podremos operar con los números comprendidos entre −32768 y 32767. Pues bien, supongamos que deseamos sumar los números binarios correspondientes a 25000 y 15000. Cada uno de ellos está dentro de los límites permitidos, pero la suma (40000) se encuentra fuera del intervalo válido. Es decir: no podremos representar el número 40000 con la precisión admitida por nuestro ordenador. Con mayor razón ocurriría algo semejante si tratáramos de multiplicar 1000 por 2000, pues el resultado es dos millones.

Para resolver estas situaciones, existen dos posibilidades. O bien buscamos un microprocesador capaz de trabajar directamente con números expresados en un sistema de representación interna llamado «coma flotante», o bien hacemos un programa que lo realice. Los dos casos se dan en los ordenadores más conocidos. En el IBM PC, por ejemplo, además del microprocesador normal de la máquina (8086, 8088 u 80286), que es capaz de realizar operaciones con enteros de ocho y dieciséis cifras binarias, se puede añadir un segundo microprocesador (co-procesador matemático 8087 u 80287), que opera con enteros de treinta y dos o sesenta y cuatro cifras binarias y con varias representaciones en coma flotante. En otras máquinas (como el Spectrum, por ejemplo) y el mismo IBM PC, si no se dispone del co-procesador matemático, existen programas que permiten trabajar con números decimales expresados en algún sistema de coma flotante. Casi todos los intérpretes y compiladores de los principales lenguajes de programación los tienen, por lo que no tenemos, en principio, que preocuparnos de ellos. Pero en algunos de estos lenguajes es importante saber en cuál de los sistemas de representación se encuentra cada una de nuestras variables, pues hay operaciones que sólo pueden realizarse en determinados casos. Volveremos sobre esto más adelante.

El sistema de coma flotante, que se utiliza también en numerosas calculadoras de bolsillo, consiste en descomponer los números en dos partes: la primera (que se llama la «mantisa») es el conjunto de las cifras del número sin tener en cuenta la posición de la coma decimal (por eso se llama «coma flotante»), que colocaremos siempre a la derecha de la primera cifra del número que sea distinta de cero, leyéndolas de izquierda a derecha. La segunda parte (el exponente) es la potencia de diez por la que habría que multiplicar la mantisa para obtener el número deseado. Por ejemplo, los tres números decimales dados anteriormente se expresarían en coma flotante así:

Número	Mantisa	Exponente
2,5	2,5	0
0,0000000001	1,0	−10
123,45	1,2345	2

En efecto: 2,5 es igual a 2,5 multiplicado por 10 elevado a cero. 0,0000000001 es igual a 1,0 multiplicado por 10 elevado a −10. Finalmente, 123,45 es igual a 1,2345 multiplicado por 10 elevado a 2, es decir, por 100. Recuérdese que 10 elevado a un número positivo «n» es igual a un uno seguido por n ceros, y que 10 elevado a un número negativo, «−n», es igual a cero, coma, n−1 ceros y un uno. Así, 10 elevado a 3 es 1000 y 10 elevado a −3 es 0,001.

De igual manera que lo hemos definido para el sistema decimal de numeración, podemos definir un sistema de coma flotante para el sistema binario, y es precisamente en esta forma como se emplea en numerosas computadoras y en los traductores de los lenguajes de programación.

APLICACIONES

TRATAMIENTO DE TEXTOS



El programa Wordstar



WORDSTAR es uno de los procesadores de textos más extendidos del mercado. La razón de esta fama está basada en dos puntos principales:

— Es un programa que está implementado en una gran variedad de ordenadores, lo cual facilita su difusión, hasta haberse convertido casi en un estándar. Específicamente existe para todos aquellos ordenadores que tengan sistema operativo CP/M o MS/DOS.

— Es muy versátil, disponiendo de una gran cantidad de comandos y órdenes con las que modificar y editar un texto lo que resuelve casi todos los problemas y formatos que se pueden presentar a la hora de crear un texto.

Una de las principales ventajas del WORDSTAR es el "formateo en pantalla", o lo que es lo mismo, el texto que estamos editando en la pantalla tiene el mismo aspecto que el que saldrá al imprimirse en papel por la impresora, y cualquier cambio que hagamos en el texto se verá reflejado de inmediato en la pantalla viendo el efecto real que produce en él.

Otra característica importante, aunque común a la mayoría de los procesadores de texto, es el "WORD WRAP", que permite escribir un texto sin preocuparse de dónde termina cada línea, ocupándose el procesador de pasar la palabra que se está escribiendo a la línea siguiente si ésta no cabe en la línea, reajustándola para que ocupe toda la longitud entre los márgenes.

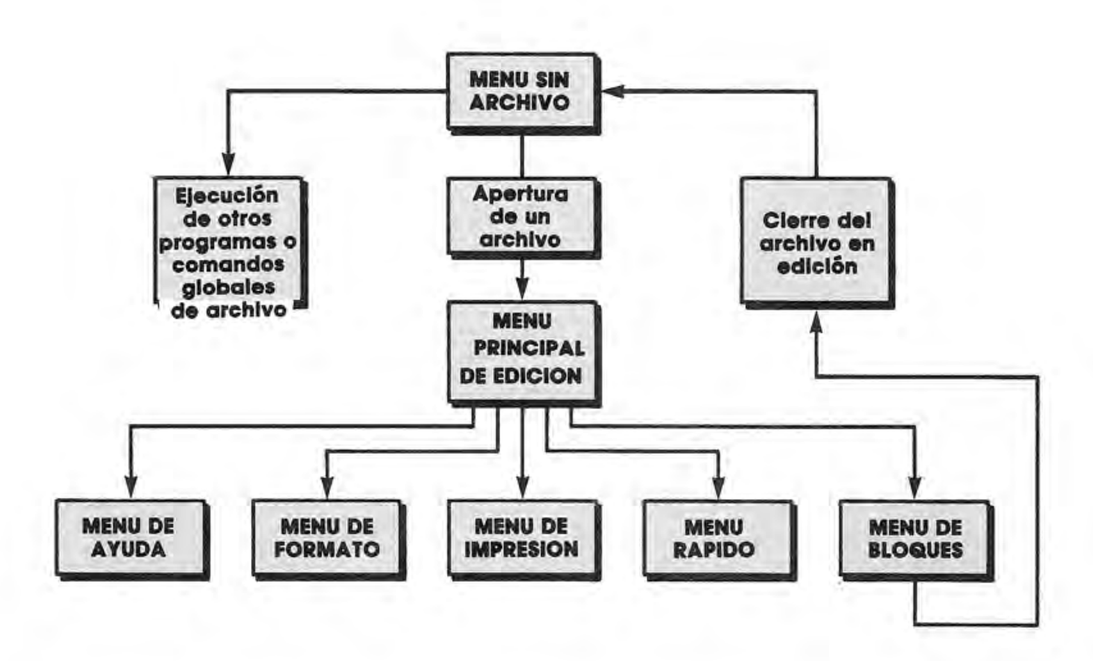
El Wordstar también permite escribir textos muy largos, ya que el almacenamiento del texto mientras se está editando se realiza en el disco, estando el tamaño del documento sólo limitado por la capacidad disponible en el disco.

Además de los comandos comunes a los procesadores de texto, dispone de una gran cantidad de funciones como: funciones de manejo de bloques, búsqueda y sustitución, paginación automática del texto, varios tipos de justificación, tabuladores decimales, varios tipos de letra en impresora, etc.



Estructura del Wordstar

Las funciones del WORDSTAR están divididas en menús que agrupan funciones similares, cuya estructura general podemos ver en la figura 1.



• Menú sin archivo

Nada más empezar con el programa entramos en el "MENU SIN ARCHIVO", en el cual podemos realizar una serie de acciones globales sobre los archivos de los documentos.

MENU SIN ARCHIVO		
Comandos Preliminares	Comandos Archivo	Comandos Sistema
L Cambiar unidad disco estándar F Dir. de disco arch. no (SI) H Fijar nivel ayuda	P IMPRIMIR arch.	R Ejec. programa X SALIR al sistema
Comandos Abrir Archivo D Abrir archivo documento N Abrir archivo no docum.	E RENOMBRAR arch. O COPIAR arch. Y BORRAR arch.	Opciones WordStar M Ejec. MailMerge S Ejec. CorrectStar

Fig. 2.

a) *Comandos preliminares*

Estos comandos nos permiten establecer el nivel de ayuda y en qué lectora de discos vamos a abrir un fichero.

b) *Comandos de apertura de archivo*

Con estas órdenes especificamos al WORDSTAR cuál va a ser el archivo con el que vamos a trabajar, abriendo un documento ya existente para corregirlo o creando uno nuevo.

c) *Comandos de manejo global de archivos*

Estos comandos permiten imprimir, renombrar, copiar o borrar un archivo cualquiera del disco.

d) *Comandos varios*

Con estas aplicaciones podemos realizar una serie de tareas no relacionadas directamente con los documentos, como ejecutar un programa externo o acabar la sesión de manejo del WORDSTAR, esto es, salir de la aplicación.

• Menú principal de edición

Una vez abierto un archivo en el menú anterior, pasamos a este menú, con el cual vamos a escribir el texto permitiendo una serie de comandos para movernos dentro y realizar borrados e inserciones.

También a partir de este menú tendremos acceso a los otros menús que nos permitirán realizar todas las funciones avanzadas del WORDSTAR.

En el menú principal podemos distinguir las siguientes partes.

A:PEPET PAG. 1 LIN. 1 COL 01 INSERTAR SI

MENU PRINCIPAL

Movimiento del cursor	Borrar	Varios	Otros menús (sólo desde pral.)
^S car. izda. ^D car. dcha.	^G car.	^I Tab. ^B Recomp.	^J Ayuda
^A pal. izda. ^F pal. dcha.	DEL car iz	^V INSERTAR SI/NO	^Q Rápido
^E lín. arr. ^X lín. abajo	^T pal dch	^L Bus./sust. otra	^O Pantalla
Deslizar:	^Y línea	RETORNO fin párrafo	^K Bloques
^Z línea abajo ^W línea arr.		^N Insertar RETORNO	^P Impresión
^C pant. arr. ^R pant. abajo		^U Cancelar comando	

Fig. 3.

a) *Comandos de desplazamiento del cursor*

Con estos comandos podemos llevar el cursor a cualquier parte del texto para visualizarlo o realizar modificaciones.

Permiten borrar caracteres, palabras o líneas del texto.

b) *Comandos varios*

Aquí se incluyen la mayoría de los comandos generales para modificar y corregir un texto, cómo reformar un párrafo, insertar caracteres en una posición, tabuladores, etc.

c) *Acceso a otros menús*

Estos comandos permiten acceder a los restantes menús dentro del modo de edición.

Los menús a los cuales podemos acudir son:

1. *Menú de ayuda*

Con este menú podemos tener en pantalla una ayuda rápida de cualquier comando del WORDSTAR, gracias a la cual no tendremos necesidad de recurrir al manual para consultar alguna duda.

MENU DE AYUDA

		Otros menús (desde menú pral.)
H Mostrar/fijar nivel ayuda	S Línea estado	^J Ayuda ^K Bloq.
B Recomp. párrafo (CONTROL-B)	R Regleta	^Q Rápido ^P Impr.
F Ind. en colum. más a derecha	M Márgenes y tabs.	^O Pantalla
D Comandos punto, contr. impr.	P Poner marcas	ESPACIO le lleva
I Indice de comandos	V Mover texto	al menú principal.

Fig. 4.

2. *Menú de manejo de bloques*

El menú de bloques permite realizar una serie de comandos relacionados con bloques de texto, como mover, copiar, borrar o almacenar el bloque. También permite el manejo de documentos completos con funciones de impresión, almacenamiento en disco, etc.

MENU DE BLOQUES

Salvar archivo	Operac. bloques	Operac. Archivos	Otros menús (desde menú pral.)
S Salv y contin.	B Princ. K final	R Leer P Imprim.	^J Ayuda ^K Bloq.
D Salv—listo	H Mostrar sí/no	O Copiar E renomb.	^Q Rápido ^P Impr.
X Salv y salir	C Copiar y borrar	J Borrar	^O Pantalla
Q Abandonar arch	V Mover W grabar	Operac. de disco	ESPACIO le lleva
Poner marcas	N Columna sí (NO)	L Cambiar disco act.	al menú principal
0-9 Ver o no 0-9		F Directoro sí (NO)	

Fig. 5.

3. *Menú rápido*

Este menú permite realizar rápidamente alguna de las funciones que aparecen en el menú principal, como movimiento con saltos más grandes, supresión o borrado, búsqueda y sustitución de palabras, repetición de órdenes, etc.

MENU RAPIDO

Movimiento de cursor	Borrar	Varios	Otros menús (desde menú pral.)
S lado. izdo. D lado dcho.	Y lín. der	F Buscar texto	^J Ayuda ^K Bloq.
E prin. pant X fondo pant.	DEL lín iz	A Buscar y sustit.	^Q Rápido ^P Impr.
R prin. arch C final arch.		L Buscar falta ort.	^O Pantalla
B prin. bloq. K final bloque		Q Repetir comando o tecla hasta que se pulse espacio	ESPACIO le lleva al menú principal.
Z abajo W arr. 0-9 marcas ref.			
P previo V últ. búsqueda o bloque			

Fig. 6.

4. *Menú de impresión*

Este menú contiene los comandos relacionados con el control de la impresora, como pausa de la impresión, salto entre líneas, y también con los efectos especiales de impresión, como negrilla, subrayado, etc.

MENU DE IMPRESION

Efectos especiales (princ. y final)	Efectos especiales (una vez de cada)	Cambios impresión	Otros menús (desde Menú pral.)
B Negr. D Doble	H Sobreimpr. carác.	A Paso alternativo	^J Ayuda ^K Bloq.
S Subrayado	O Esp. irrompible	N Paso estándar	^Q Rápido ^P Impr.
X Tachado	F Espacio fantasma	C Pausa en impresión	^O Pantalla
V Subíndice	G Borrado fantasma	Y Cinta otro color	ESPACIO le lleva al menú principal.
T Sobreíndice	RET Sobreimp. línea	Parches usuario Q(1) W(2) E(3) R(4)	

Fig. 7.

5. *Menú de formato en pantalla*

Los comandos agrupados en este menú permiten el control del aspecto de un documento variando su formato. Permite elegir el tipo de justificación, la posición de los tabuladores y márgenes, control de fin de página, etc.

MENU PANTALLA

Margen y tabs.	Funciones línea	Más cambiadores	Otros menús
L Poner marg. izdo.	C Centras texto	J Justific. no (SI)	^J Ayuda ^K Bloq.
R Poner marg. dcho.	S Poner esp. línea	V Vari-tabs no (SI)	^Q Rápido ^P Impr.
X Liberar márgenes		H Ay. guión no (SI)	^O Pantalla
I Fij. N Borr. tab.	Cambiadores	E Guión aut. sí (NO)	ESPACIO le lleva al menú principal.
G Tab. párrafo	W Trans. pal. no (SI)	D Ver impr. no (SI)	
F Regleta des. lín.	T Regl. lín. no (SI)	P Sep. pág. no (SI)	

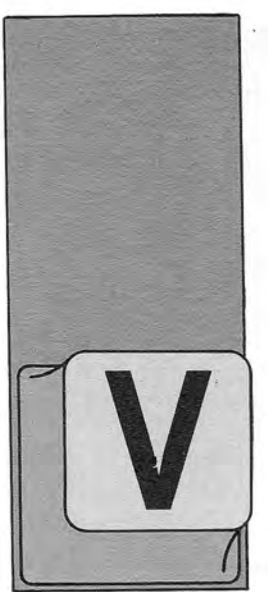
Fig. 8.

Aparte de todos los comandos accesibles por medio de los menús para el control del formato del texto, existe una serie de órdenes que van incluidas dentro del texto y que nos permiten controlar algunas características de la impresión como el formato general de la página, que incluye márgenes superior e inferior, e izquierdo y derecho. Estos comandos, aunque vayan incluidos en el texto, no se imprimen y sólo sirven para controlar la impresión.

Para que el WORDSTAR reconozca estos comandos, todos ellas van precedidos por un punto (.). Y tienen que estar en una línea cada uno, sin más texto detrás.

PASCAL

LA ESTRUCTURA IF-THEN (continuación)



VAMOS a comentar un detalle un poco delicado de las estructuras IF-THEN-ELSE; si no se ve muy claro ahora, no hay que alarmarse, pues es del tipo de cosas que se van entendiendo con la práctica.

En el último programa de ejemplo vimos cómo se podían poner estructuras IF dentro de estructuras IF. En estas situaciones, si el compilador llegase a la palabra reservada ELSE, supondría que corresponde a la última estructura IF que no la tuviera todavía. Por ejemplo:

```
if Alto then  (* Si alto, mirar si es bueno *)

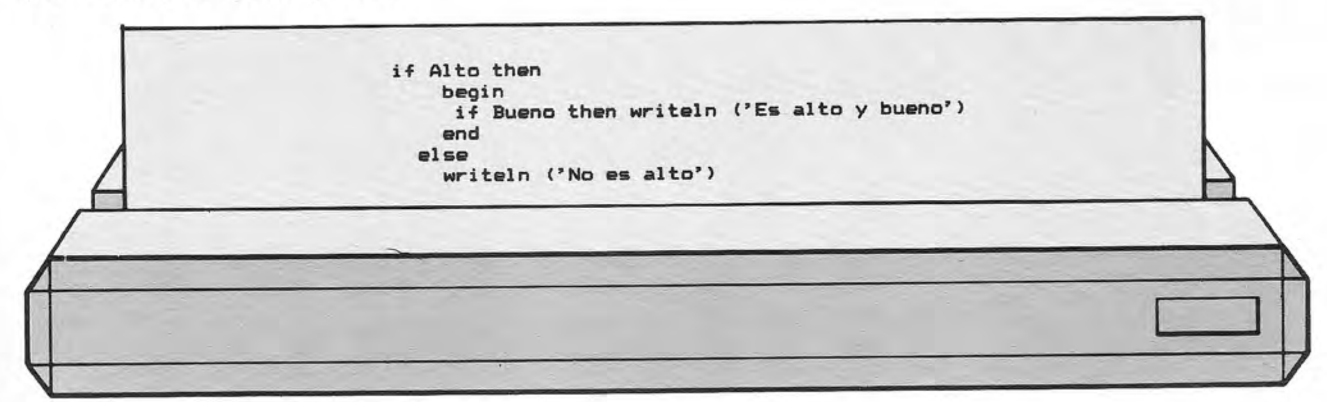
    if Bueno then
        writeln ('Es alto y bueno')
      else
        writeln ('Es alto pero no bueno')

  else (* Si no es alto, mirar si es listo *)

    if Listo then
        writeln ('No es alto, pero es listo')
      else
        writeln ('No es alto ni listo')
```

Al llegar al primer ELSE de todos, el compilador sobreentiende que corresponde al último IF sin ELSE, es decir, a «if Bueno...»; al llegar al segundo ELSE, el último IF sin ELSE que queda es «if Alto...» y, por tanto, le corresponde.

Si quisiésemos tener una estructura IF simple dentro de otra con ELSE, podríamos hacer lo siguiente:



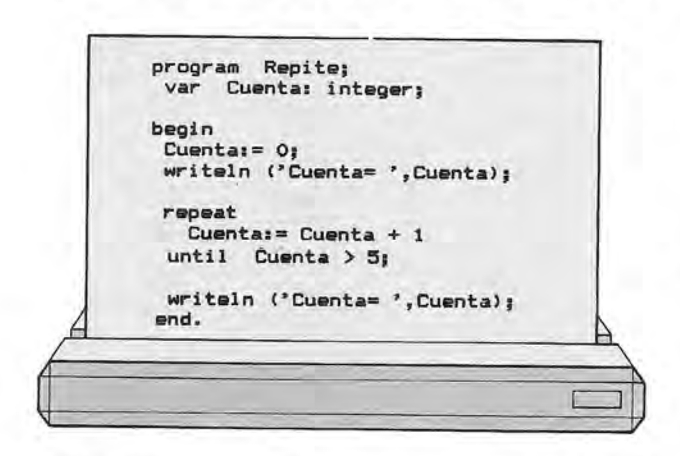
```
if Alto then
    begin
      if Bueno then writeln ('Es alto y bueno')
    end
  else
    writeln ('No es alto')
```

Utilizando una secuencia de una sola instrucción (lo que en principio podría parecer absurdo), las palabras BEGIN y END han hecho de «paréntesis» para así forzar a que ELSE corresponda a la primera estructura IF.



La estructura REPEAT

Hay ocasiones en que se necesita repetir la ejecución de una cierta instrucción hasta que se cumpla una condición dada. Para ello se dispone de la estructura REPEAT. Empecemos con un ejemplo:



Si ejecutamos este programa, en la pantalla del ordenador aparecerá:



La estructura REPEAT se compone de las palabras reservadas REPEAT y UNTIL, la instrucción a repetir escrita entre ellas, y, en último lugar, la condición por la que se debe terminar con las repeticiones y pasar a lo siguiente.

Durante la ejecución del programa, al llegarse a la estructura REPEAT se ejecuta la instrucción que alberga, tras lo cual se pasa a evaluar la condición que hay tras la palabra reservada UNTIL. Si el resultado es TRUE, se pasa a lo que hubiera a continuación, pero si fuera FALSE, se volvería a ejecutar la instrucción anterior y a evaluar la condición, etc., hasta que en algún momento, por fin, el resultado fuese TRUE.

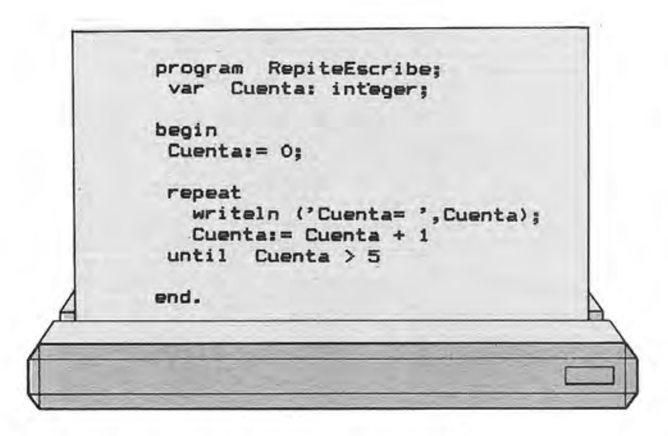
Por tanto, la primera vez que se ejecuta la instrucción. Cuenta: = Cuenta + 1, la variable Cuenta, que valía 0, pasa a valer 1. Tras eso, el resultado de la expresión Cuenta > 5 es, por supuesto, FALSE, por lo que se volverá a ejecutar Cuenta: = = Cuenta + 1 (Cuenta valdrá entonces 2), a evaluar Cuenta > 5, etc.

Por fin, llegará un momento en que Cuenta valdrá 5; tras ejecutarse Cuenta: = = Cuenta + 1, pasará a valer 6 y entonces al evaluarse la Cuenta > 5 el resultado será TRUE, por lo que se pasará a la instrucción WRITELN que hay a continuación. Traduciendo del inglés resulta todo muy claro:

REPEAT	Cuenta:=Cuenta+1	UNTIL	Cuenta > 5
Repetir	Cuenta:=Cuenta+1	hasta que	Cuenta mayor que 5

La instrucción a repetir PUEDE SER DE CUALQUIER TIPO, simple o estructurada.

Cuando la instrucción es una secuencia, como aquí hay dos palabras reservadas, REPEAT y UNTIL, por delante y detrás de ella, no hace falta poner BEGIN y END para delimitarla. Por tanto, la o las instrucciones a repetir se escriben una detrás de otra separadas por punto y coma entre las palabras REPEAT y UNTIL. Probemos un programa parecido al anterior:



En la pantalla, aparecerá:



A las situaciones en que un conjunto de instrucciones se repite una y otra vez se les llama «bucles de programa».

La condición de salida del bucle (o sea, la que decide si hay que seguir repitiendo o no) puede ser cualquier expresión que dé un resultado de tipo BOOLEAN.

El conjunto formado por la palabra REPEAT, la o las instrucciones, la palabra UNTIL y la expresión lógica es a su vez una instrucción estructurada (cuya ejecucción consiste en repetir las que contiene hasta que se cumpla la condición) y, por tanto, se le aplica todo lo dicho hasta el momento sobre éstas.

Con REPEAT podemos controlar los errores al teclear datos mejor que como lo hicimos en el programa Secuencia:

```
program PideInicial;

var
  Inicial: char;
  EsLetra: boolean;

begin
 repeat (* Repetir pregunta hasta respuesta buena *)
   writeln ('Inicial? ');  readln (Inicial);
   EsLetra:= (Inicial >= 'A') and (Inicial <= 'Z');
   if not EsLetra then
        writeln ('Imposible, repita.')
 until  EsLetra;

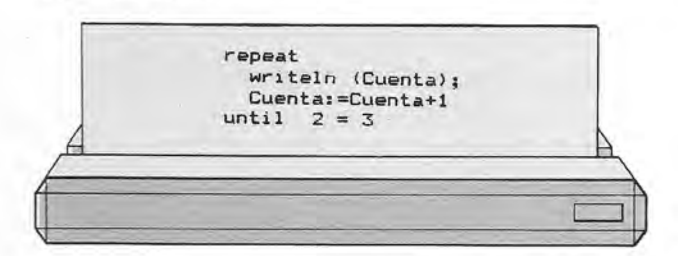
 writeln('Letra= ',Inicial)
end.
```

Gracias a la estructura REPEAT, se pedirá la inicial una y otra vez, hasta que sea correcta.

Se ha utilizado la variable EsLetra porque, como la corrección de la letra hay que analizarla para avisar en su caso (estructura IF), si guardamos el resultado de la comparación en ella se evita tener que repetirla tras UNTIL.

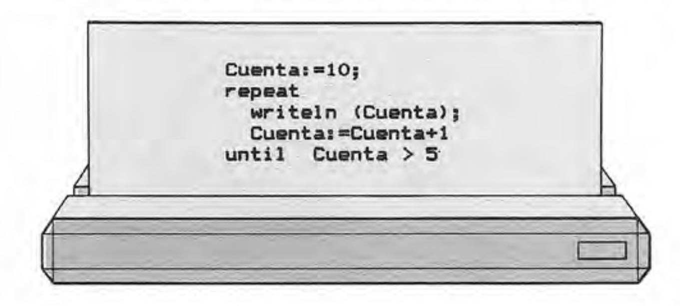
Hay que tener en cuenta dos importantes aspectos de la estructura REPEAT:

1. Si el bloque de instrucciones a repetir no afectara para nada a la condición de salida, podríamos tener un «bucle infinito»:



La condición de salida (2=3) nunca se cumple y por más que se repitan una y otra vez las dos instrucciones del bucle, nunca se cumplirá, con lo que la ejecución del programa no avanzará nunca: tenemos un bucle infinito. (Por cierto, si en algún momento se deseara tener un bucle infinito deliberadamente, lo mejor es escribir REPEAT ... UNTIL FALSE.)

2. La condición de salida (la expresión BOOLEAN) se evalúa DESPUES de la ejecución de la o las instrucciones del bucle. Por ello, aunque el valor fuera TRUE ya desde un principio, siempre se ejecutarán las instrucciones al menos una vez:



Con estas instrucciones aparecería 10 en la pantalla, aunque la condición para no seguir repitiendo el bucle es que Cuenta sea mayor que cinco.

Para los casos en que se deba comprobar la condición de salida ANTES de ejecutar las instrucciones del bucle una sola vez, se dispone de la estructura WHILE, que veremos a continuación.



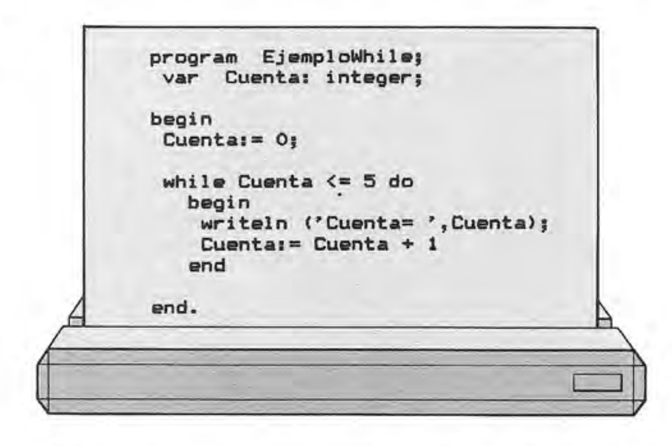
La estructura WHILE

Esta estructura es similar a la instrucción REPEAT, con dos diferencias:

1. La expresión que controla si se debe seguir con las repeticiones o no, se evalúa ANTES de ejecutar las instrucciones del bucle.

2. Se sale del bucle cuando la condición deja de cumplirse, es decir, cuando el resultado de la expresión de control es FALSE.

Veamos otro ejemplo para ilustrar esto:



Este programa actúa igual que RepiteEscribe. Una vez más, la traducción del inglés resulta clarificadora.

WHILE	Cuenta <= 5	DO	...
Mientras	Cuenta menor o igual que 5	hacer	(la instrucción)

Es decir, entre las palabras reservadas WHILE y DO se escribe la condición de control (una expresión cuyo resultado sea de tipo BOOLEAN) y tras todo ello, la

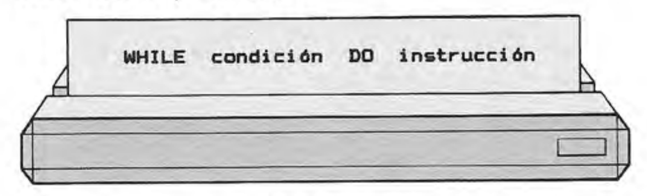
instrucción a repetir, que puede ser cualquiera, estructurada o no (en el ejemplo es una secuencia).

Durante la ejecución del programa, al llegarse a una estructura WHILE se evalúa la condición, y si el resultado fuera TRUE, se pasaría a ejecutar la instrucción a repetir. Tras ello, se volvería a evaluar la condición y a ejecutar la instrucción caso de que el resultado hubiese sido nuevamente TRUE, etc., hasta que llegase un momento en que el resultado fuera FALSE, en cuyo caso se continuaría la ejecución del programa con la instrucción que hubiese a continuación de la estructura.

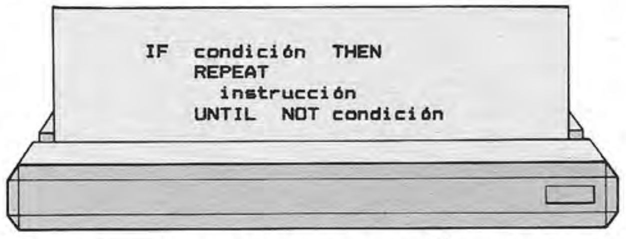
Por ello, si en el ejemplo se cambiara la instrucción Cuenta: = 0 por Cuenta: = 10, nunca se llegaría a ejecutar el bucle, pues ya la primera vez el resultado de la condición sería FALSE.

Nuevamente la estructura en su conjunto es toda ella una instrucción estructurada.

NOTA: Si se desea profundizar en estas cuesitones se puede observar que la instrucción WHILE equivale a una combinación de IF y REPEAT:



(«mientras se dé la condición haz la instrucción») equivale a:



(o sea, «si se da la condición, repite la instrucción hasta que no se dé»).

Para terminar, veamos un programa que calcula y presenta todas las potencias de dos menores que un número dado:

```
program  PotenciasDeDos;

 var
   N, Tope  : integer;
   Error,
   CabeComoEntero: boolean;

 (* CabeComoEntero sirve para indicar si al pasar a la
 siguiente potencia nos saldremos de los límites INTEGER *)

begin
  N:= 2;                    (* 2 es la primera potencia *)

  (* Pedir límite hasta que sea correcto: *)
  repeat
    write ('Límite: '); readln (Tope);
    Error:= (Tope < 0);     (* Error si límite negativo *)
    if Error then writeln ('No vale. Repita.')
  until not Error;

  CabeComoEntero:= true;

  while (N <= Tope) and CabeComoEntero do
    begin
     writeln (N:6);

     CabeComoEntero:= (N <= MaxInt div 2);
  (* Si es TRUE, el siguiente valor es <= que MaxInt *)

     if CabeComoEntero then N:= N * 2
  (* Cada potencia es igual al doble de la anterior. *)

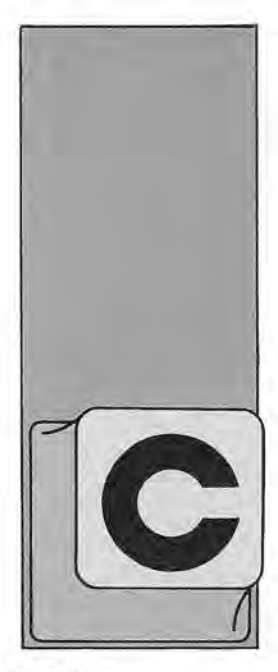
    end;

  writeln ('Se acabó.')
end.
```

El programa lo que tiene que hacer es sacar la primera potencia (2) y calcular la siguiente (4); tras esto, debe sacar la última calculada (4) y calcular la siguiente (8). Todo este proceso debe REPETIRSE hasta alcanzar el tope o no poder seguir debido a las limitaciones inherentes al tipo INTEGER utilizado. Si Tope fuera 1, no se debería sacar ni la primera potencia, de ahí el empleo de WHILE en vez de REPEAT.

Se ha utilizado la variable CabeComoEntero para evitar calcular un valor que se salga de los límites admitidos para los números enteros.

Como ejercicio, podríamos modificar el programa para obtener las potencias de tres (o mejor, utilizar una constante declarada en lugar de 2 para hacerse el mínimo de cambios cada vez). Asimismo, podríamos utilizar una estructura REPEAT y ver la diferencia cuando Tope valga 1.



OTROS LENGUAJES

SISTEMAS OPERATIVOS: CP/M



Introducción

CP/M es un sistema operativo monousuario y monotarea orientado inicialmente a equipos basados en los microprocesadores Intel 8080 y 8085, así como Zilog Z-80.

CP/M son las iniciales de Control Program for Microprocessors (programa de control para microprocesadores). Fue escrito por Gary Kildall en 1973, cuando trabajaba para la compañía Intel, al verse en la necesidad de trabajar con discos flexibles y no tener ningún programa que controlase algunas interacciones entre el microprocesador y los disquetes. La compañía Intel rechazó el proyecto de Kildall de continuar desarrollando este incipiente sistema operativo, por lo que fundó su propia empresa (la que hoy es Digital Research Corporation).

En 1976 Zilog lanza al mercado el microprocesador Z-80, y Kildall crea para éste la versión 1.4 de CP/M, que es la primera versión industrial del programa. Esta versión podía trabajar en sistemas que tuvieran sólo 16 Kbytes de memoria central, ocupando muy poca de esa memoria.

El éxito obtenido por el microprocesador Z-80 hizo que CP/M se hiciera muy popular. Al mismo tiempo, la biblioteca de programas creados para este sistema operativo iba en aumento, lo que también influyó para que los constructores de equipos adoptasen CP/M como sistema operativo para sus máquinas, ya que ello significaba disponer para las mismas de gran cantidad de software nada más sacarlas al mercado.

En la actualidad, CP/M es el sistema operativo indiscutible para equipos basados en microprocesadores de 8 bits. En el ámbito de los 16 bits, sin embargo, se ha visto totalmente superado por el MS/DOS, en gran parte por la adopción de este último por parte de IBM para su IBM-PC.



Estructura de CP/M

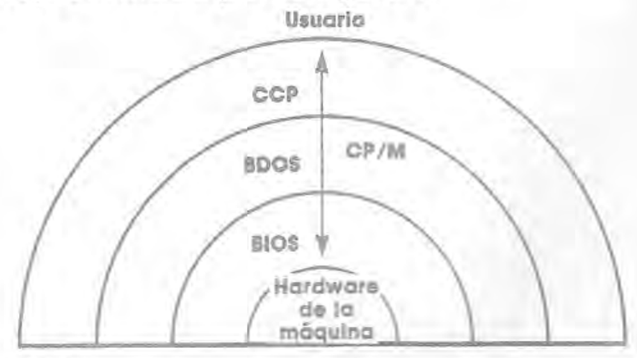
CP/M se divide en 3 módulos principales:

— BIOS (Basic Input Output System): Es el programa que interactúa directamente con el hardware de la máquina. Maneja la consola (teclado y pantalla), el disco, la impresora y cualquier otro dispositivo periférico conectado al ordenador.

— BDOS (Basic Disk Operating System): Es el módulo encargado del manejo del disco en lo relativo a ficheros. Controla el directorio de ficheros, permite la ubicación dinámica de espacio, etc.

— CCP (Console Command Processor): Es un intérprete de comandos, y sirve como interface o "mediador" entre el usuario y el resto del sistema operativo. Lee las instrucciones tecleadas por el usuario y dirige a los demás módulos para la realización de las acciones pertinentes.

Una visión por capas de CP/M sería la representada por la figura 1:

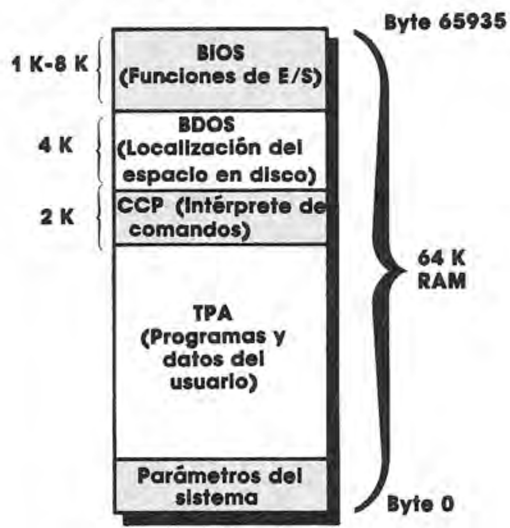




Estructura por capas de CP/M.

Otra parte fundamental de CP/M, aunque no es un programa, es la TPA (Transit Program Area) o área de programas transitorios. Es la zona de memoria donde CP/M conserva los programas cargados desde disco, con sus datos correspondientes.

Para ver cómo se instala CP/M en memoria RAM, supongamos una memoria de 64 Kbytes (que es lo típico). En las direcciones más bajas se cargan los parámetros del sistema (rutina de arranque, etcétera). Luego viene la TPA, que es lo que más espacio ocupa (mientras mayor sea ésta, mayor espacio libre tiene el usuario para sus programas). Por último, en las direcciones más altas se carga el CCP, el BDOS y el BIOS.




Mapa de memoria típico de CP/M.



Comandos del CP/M

Al igual que MS/DOS, CP/M tiene algunos comandos internos que están siempre en memoria RAM, en la zona ocupada por el CCP.

Estos son los siguientes:

— **DIR:** Visualiza por pantalla el directorio de un disco.

— **ERA:** Borra un fichero almacenado en disco.

— **SAVE:** Almacena en disco el contenido de un fichero que está en memoria.

— **TYPE:** Visualiza por pantalla el contenido de un disco.

— **REN:** Cambia el nombre de un fichero en disco.

— **USER:** Define áreas de usuario en el directorio.

Los comandos externos de CP/M residen en disco en forma de ficheros con la extensión ".COM", y se cargan en la zona más baja de la TPA cuando se les invoca. Algunos de estos comandos son los siguientes:

— **PIP:** Es un programa de intercambio entre periféricos. Realiza las funciones básicas de copia, carga, impresión y combinación de ficheros.

— **STAT:** Proporciona diversa información sobre el espacio ocupado en disco y la asignación de los dispositivos.

— **LOAD:** Carga ficheros en código hexadecimal y produce ficheros ejecutables.

— **SUBMIT:** Carga en memoria un fichero de comandos para procesarlos en modo "batch".

— **SYSGEN:** Inicializa un disquete y graba el CP/M en él.

— **ED:** Es un editor de líneas, útil en la creación y modificación de ficheros.



Familia CP/M

Bajo el nombre de CP/M realmente se engloba a toda una familia de sistemas operativos y versiones, nacidas todas ellas de un tronco común, pero adaptadas cada una a distintos entornos y sistemas microinformáticos.

Básicamente, la familia CP/M se compone de los siguientes miembros:

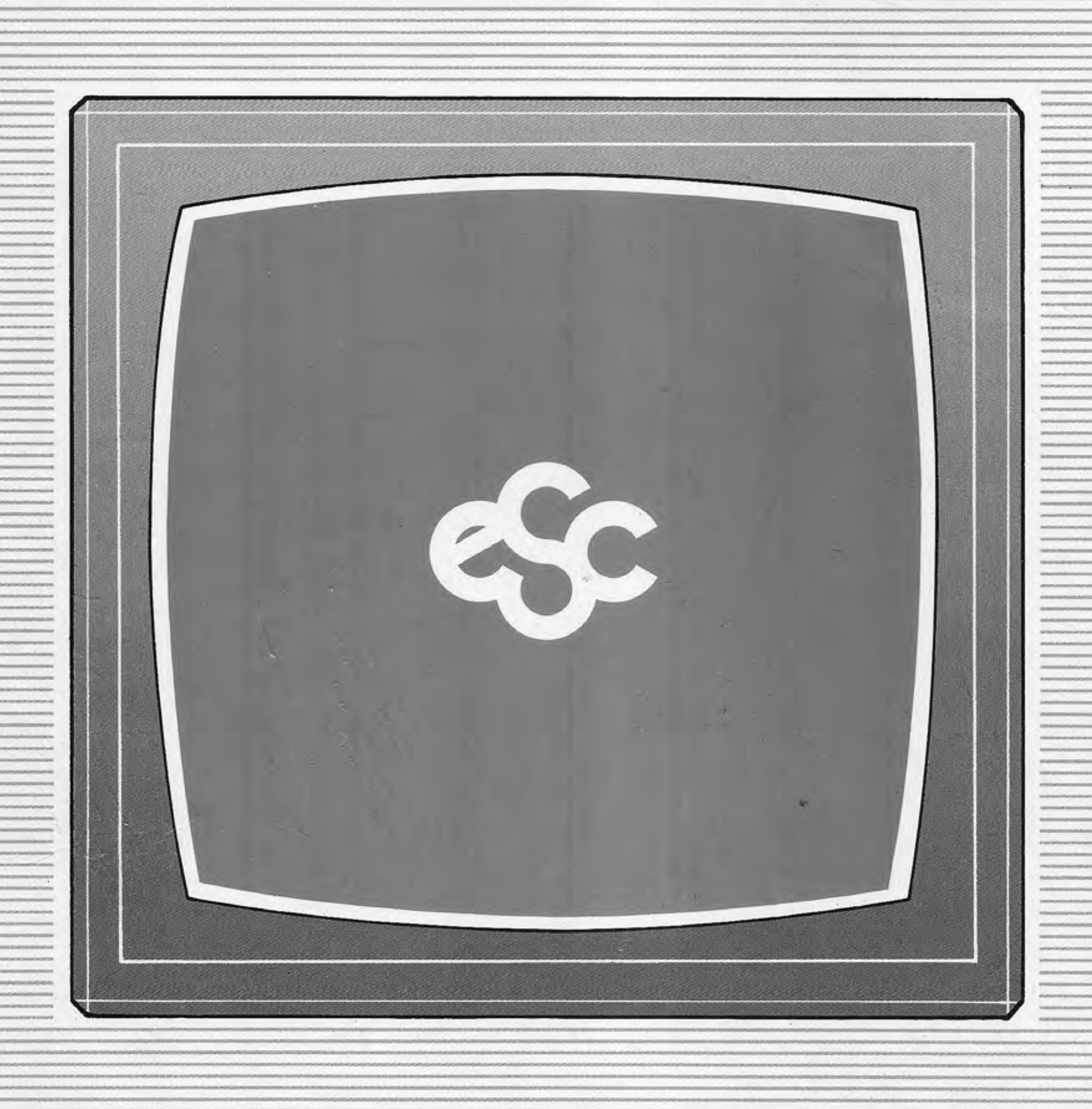
— CP/M 80: Versión monousuario y monotarea para sistemas basados en los microprocesadores de 8 bits 8080, 8085 y Z-80.

— CP/M 86: Análoga a la anterior, pero para los microprocesadores de 16 bits 8088 y 8086.

— CP/M 68K: Versión adaptada al microprocesador MC68000.

— CP/M 86 concurrente: Versión multitarea del CP/M 86.

— MP/M 80 y MP/M 86: Versiones multiusuario del CP/M 80 y del CP/M 86, respectivamente.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼